MuSC: Improving Complex Instruction Following with Multi-granularity Self-Contrastive Training

Hui Huang[1]*, **Jiaheng Liu**[2]*, **Yancheng He**[1]*, **Shilong Li**[3], **Bing Xu**[1], **Conghui Zhu**[1], **Muyun Yang**[1]✉, **Tiejun Zhao**[1]

[1]Faculty of Computing, Harbin Institute of Technology, [2]Nanjing University
[3]School of Artificial Intelligence, Beijing University of Posts and Telecommunications
huanghui@stu.hit.edu.cn, yangmuyun@hit.edu.cn

Abstract

Complex instruction-following with elaborate constraints is imperative for Large Language Models (LLMs). While existing methods have constructed data for complex instruction alignment, they all rely on a more advanced model, especially GPT-4, limiting their application. In this paper, we propose a Multi-granularity Self-Contrastive Training (MuSC) framework, to improve the complex instruction alignment without relying on a stronger model. Our method is conducted on both coarse and fine granularity. On coarse-granularity, we construct constraint-aware preference data based on instruction decomposition and recombination. On fine-granularity, we perform token-aware preference optimization with dynamic token-level supervision. Our method is evaluated on open-sourced models, and experiment results show our method achieves significant improvement on both complex and general instruction-following benchmarks, surpassing previous self-alignment methods[1].

1 Introduction

Large Language Models (LLMs) have made remarkable advancements and are being wildly applied across various domains (Zhao et al., 2024; Wu et al., 2024a; He et al., 2024c; Liu et al., 2025; He et al., 2024b; Zhang et al., 2025). The instruction-following ability is fundamental and important, as it enables LLMs to generate appropriate responses to given instructions and solve corresponding tasks (OpenAI et al., 2024). While recent LLMs perform comparatively well on simple instructions, their response quality to complex instructions with elaborate constraints often falls under expectation, with some of the constraints omitted (He et al., 2024a; Jiang et al., 2024b), which hinders their application in more real-world complex scenarios.

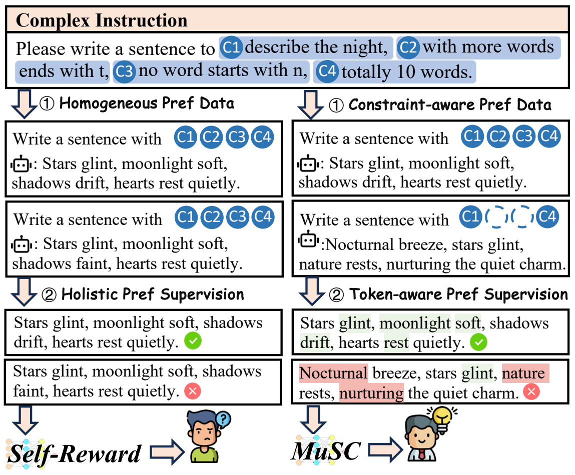

Complex Instruction
Please write a sentence to C1 describe the night, C2 with more words ends with t, C3 no word starts with n, C4 totally 10 words.
① Homogeneous Pref Data
① Constraint-aware Pref Data
Write a sentence with C1 C2 C3 C4
: Stars glint, moonlight soft, shadows drift, hearts rest quietly.
: Stars glint, moonlight soft, shadows faint, hearts rest quietly.
Write a sentence with C1 C4
:Nocturnal breeze, stars glint, nature rests, nurturing the quiet charm.
② Holistic Pref Supervision
② Token-aware Pref Supervision
Self-Reward
MuSC

Figure 1: An illustrative comparison between our method and Self-Reward. Note that Self-Reward cannot create effective contrast for complex instruction-following, resulting in suboptimal optimization.

To enhance the complex instruction following, the core challenge is the scarcity of high-quality complex instruction data (Lou et al., 2024). Most existing instruction datasets are constructed based on existing NLP datasets or question-answering websites with simple constraints (Wang et al., 2023; Taori et al., 2023; Lian et al., 2023; Longpre et al., 2023). To cope with the scarcity of complex instruction data, previous work such as Evol-Instruct (Xu et al., 2024), Conifer (Sun et al., 2024a), and Self-Correct (Palmeira Ferraz et al., 2024) have been proposed to construct complex instructions and responses. However, *these methods typically rely on a high-performance proprietary model (e.g., GPT-4) to distill the complex instruction-following ability, which is expensive and can not be scaled up in real-world applications.*

Recently, the research community has paid attention to self-alignment, to break the data bottleneck without relying on a stronger model (Wang et al., 2024; Zhang et al., 2024a). Self-Reward

∗ Equal contribution. ✉Corresponding Author.
[1]Codes are openly available at https://github.com/HuihuiChyan/MuSC.

(Yuan et al., 2024) proposes to utilize the model itself to both generate responses and evaluate the results, which can be incorporated into DPO training. ISHEEP (Liang et al., 2024) proposes an automatic loop to self-assess and self-filter instruction data. *Despite their effectiveness, these methods are targeted at general instruction-following ability. The self-alignment of complex instruction-following ability remains unexplored.*

In this paper, to address the above limitations, we propose a novel **Mu**lti-granularity **S**elf-**C**ontrastive Training framework (**MuSC**) in Figure 1, which mainly comprises the following components:

1) **Coarse-grained Contrast: Constraint-aware Preference Data Construction**. To improve the model's comprehension of constraint-level distinctions, we construct preference pairs that reflect the disparities in constraint fulfillment. We achieve this by breaking down each complex instruction into atomic constraints and selectively omitting a subset to form negative instructions. The chosen response, derived from the original instruction, is paired with the rejected response, generated from the negative instruction, as a contrastive pair. Notably, no external models are utilized in this construction process.

2) **Fine-grained Contrast: Token-aware Preference Optimization**. For complex instructions, the responses often involve multiple tokens that contribute differently to fulfilling the instruction's constraints. Therefore, we introduce a token-aware optimization framework that integrates dynamic token-level weights based on the model's confidence. By focusing on tokens that deviate from the constraints, this approach effectively identifies and corrects tokens where the model fails to satisfy the instruction's requirements, leading to more contextually appropriate responses.

Moreover, we need to mention that our MuSC can be applied on both pre-existing complex instruction datasets, or newly generated instruction datasets created by data synthesis methods (e.g., Self-Instruct (Wang et al., 2022)).

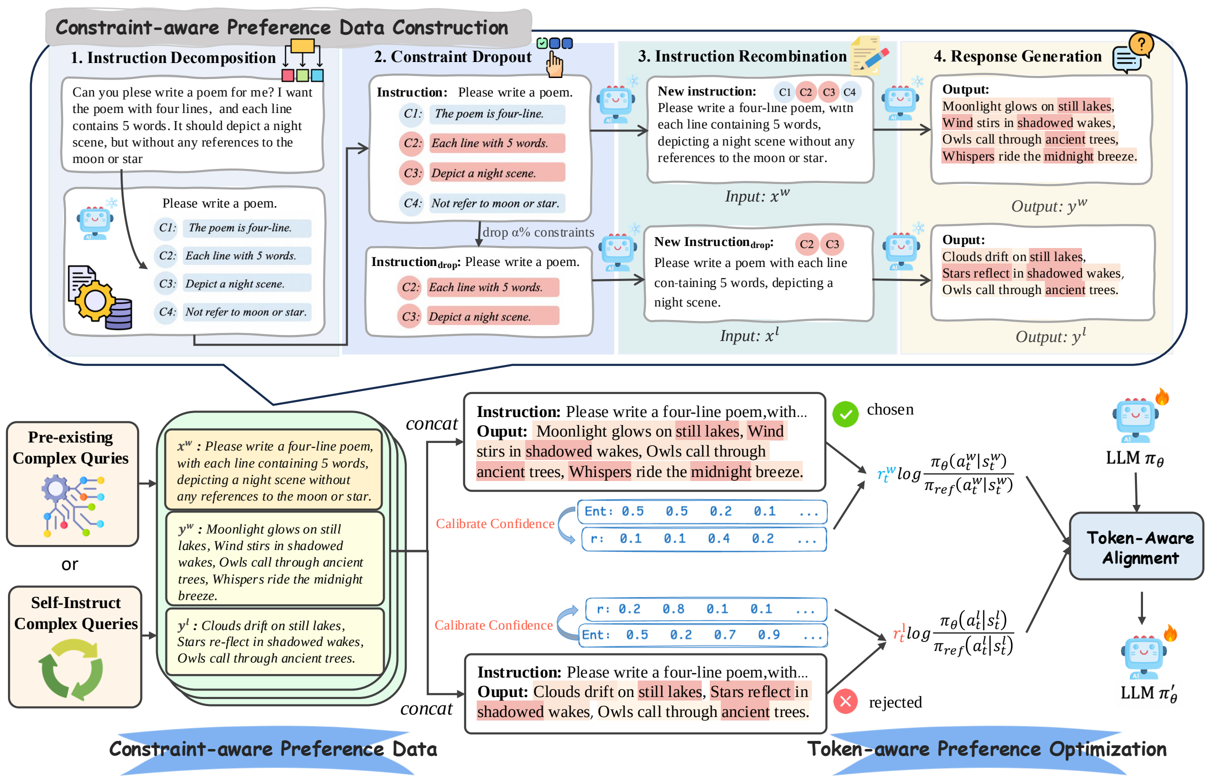
Our contribution can be summarized as follows:

- We propose a novel Multi-granularity Self-Contrastive Training (MuSC) framework, which creates effective contrast on both coarse and fine granularity, to enhance the complex instruction following abilities.
- For coarse-grained contrast, we construct constraint-aware preference data with instruction decomposition-recombination. For fine-grained contrast, we adopt dynamic token-level weight with confidence guidance for better preference optimization.
- We evaluate our framework on open-source LLMs, and achieve significant improvements on both complex and general instruction following benchmarks, without the help of a larger model or human supervision.

2 Related Work

Complex Instruction-Following. As one of the cores of LLM intelligence, how to improve the model's instruction-following capability is important. The earliest works, such as Alpaca (Taori et al., 2023), Vicuna (Chiang et al., 2023), and Camel (Wang et al., 2023), used instruction data generated by proprietary models to supervise fine-tuning of open-source models, significantly enhancing their instruction-following capabilities. However, these methods mainly focus on general instruction following, while complex instruction following still remains challenging. To cope with this challenge, a lot of methods (Yin et al., 2023; Lou et al., 2023; He et al., 2024a; Sun et al., 2024b; Chen et al., 2024b; Dong et al., 2024) have been proposed to construct complex instruction data. The earliest work is Evol-Instruct (Xu et al., 2024), which proposed to utilize GPT-4 to expand the instructions from both depth and width, thereby generating complex instructions and corresponding constraints. Conifer (Sun et al., 2024a) proposed a progressive learning strategy designed to help smaller models incrementally enhance their abilities.

Self-Alignment. Self-alignment refers to aligning the model to human preference without relying on a more advanced model or external supervision. As an early study, Self-Rewarding (Yuan et al., 2024) proposed the model itself to both generate responses and evaluate the results. Following this work, many works (Liu et al., 2024; Chen et al., 2024b,a; Pang et al., 2024; Meng et al., 2024) are conducted to obtain supervision data by the model itself. Meta-Rewarding (Wu et al., 2024b) advanced the concept by improving the model's instruction and evaluation capabilities simultaneously. Liu et al. (2024) employed diverse prompts to guide the model to generate various responses. Despite the progress these methods have made, they all target general instruction following. For



Constraint-aware Preference Data Construction
1. Instruction Decomposition
Can you plese write a poem for me? I want the poem with four lines, and each line contains 5 words. It should depict a night scene, but without any references to the moon or star
Please write a poem.
C1: The poem is four-line.
C2: Each line with 5 words.
C3: Depict a night scene.
C4: Not refer to moon or star.
2. Constraint Dropout
Instruction: Please write a poem.
drop α% constraints
Instruction$_{drop}$: Please write a poem.
3. Instruction Recombination
New instruction: Please write a four-line poem, with each line containing 5 words, depicting a night scene without any references to the moon or star.
Input: x^w
New Instruction$_{drop}$: Please write a poem with each line con-taining 5 words, depicting a night scene.
Input: x^l
4. Response Generation
Output: Moonlight glows on still lakes, Wind stirs in shadowed wakes, Owls call through ancient trees, Whispers ride the midnight breeze.
Output: y^w
Ouput: Clouds drift on still lakes, Stars reflect in shadowed wakes, Owls call through ancient trees.
Output: y^l
Pre-existing Complex Quries
or
Self-Instruct Complex Queries
x^w : Please write a four-line poem, with each line containing 5 words, depicting a night scene without any references to the moon or star.
y^w : Moonlight glows on still lakes, Wind stirs in shadowed wakes, Owls call through ancient trees, Whispers ride the midnight breeze.
y^l : Clouds drift on still lakes, Stars re-flect in shadowed wakes, Owls call through ancient trees.
concat
Instruction: Please write a four-line poem,with...
Ouput: Moonlight glows on still lakes, Wind stirs in shadowed wakes, Owls call through ancient trees, Whispers ride the midnight breeze.
Calibrate Confidence
Ent: 0.5 0.5 0.2 0.1 ...
r: 0.1 0.1 0.4 0.2 ...
r: 0.2 0.8 0.1 0.1 ...
Ent: 0.5 0.2 0.7 0.9 ...
Instruction: Please write a four-line poem,with...
Ouput: Clouds drift on still lakes, Stars reflect in shadowed wakes, Owls call through ancient trees.
chosen
rejected
$r_t^w \log \frac{\pi_\theta(a_t^w|s_t^w)}{\pi_{ref}(a_t^w|s_t^w)}$
$r_t^l \log \frac{\pi_\theta(a_t^l|s_t^l)}{\pi_{ref}(a_t^l|s_t^l)}$
LLM π_θ
Token-Aware Alignment
LLM π'_θ
Constraint-aware Preference Data
Token-aware Preference Optimization

Figure 2: The pipeline of our proposed MuSC. The process starts with constraint-aware preference data construction, which includes instruction decomposition, constraint dropout, instruction recombination and response generation. Next, the token-aware DPO is performed based on calibrated confidence to achieve token-level alignment.

complex instructions with multiple constraints, the response will be lengthy and multi-facet, resulting in challenges for the self-evaluation process.

3 Approach

The pipeline of MuSC is shown in Figure 2.

3.1 Constraint-aware Preference Data

Reinforcement-learning methods, such as PPO (Schulman et al., 2017) and DPO (Rafailov et al., 2024b), have achieved notable success in LLM optimization. Research has shown that learning from negative samples is significantly more efficient than learning solely from positive samples (Yang et al., 2024b). However, these methods are limited by the need for high-quality preference data, which is particularly scarce for complex instructions.

To construct effective preference data for complex instruction following, we propose a novel data construction method, with the following steps[1]:

1. Instruction Decomposition: A complex instruction is typically a combination of multiple atomic constraints. We decompose the complex instruction into individual atomic constraints, denoted as `Cons`.
2. Constraint Dropout: From the decomposed constraints `Cons`, we randomly eliminate $\alpha\%$ of the constraints to form $\texttt{Cons}_{drop}$.
3. Instruction Recombination: We recombine both the original and the dropped constraints `Cons` and $\texttt{Cons}_{drop}$, to create chosen and rejected instructions: `Ins` and $\texttt{Ins}_{drop}$.
4. Response Generation: Based on `Ins` and $\texttt{Ins}_{drop}$, we generate the chosen response `Resp` and the rejected response $\texttt{Resp}_{drop}$.

Previous research has suggested that the construction of effective preference pairs for optimization is non-trivial (Ivison et al., 2024). Our data construction pipeline is guided by three principles:

- **Negativity**: The rejected response should deviate from the instruction by omitting some constraints. Our method generates the rejected instruction based on corrupted constraints, ensuring that the rejected response deviates from the original complex instruction.
- **Consistency**: The rejected response should reside within the model's decoding space (Guo

[1]Please refer to Appendix A.1 for implementation details.

et al., 2024). In our method, the rejected instruction is simply a recombination of the original instructions, ensuring the response falls within the decoding space, which is crucial for the optimization process.

- **Contrastiveness**: Chosen and rejected responses should be with a rational edit distance, to form an effective contrast (Jiang et al., 2024a). By reconstructing both chosen and rejected instructions using the same method, we ensure that the derived samples do not deviate too far from each other.

With constructed data satisfying both **negativity**, **consistency** and **contrastiveness**, we form a solid foundation for effective alignment. Moreover, our method does not require a stronger model or human supervision, ensuring its scalability.

Our self-construction method can be applied in different scenarios. On one hand, it can be directly applied on pre-existing complex instruction dataset. On the other hand, if there is no existing complex queries, we can adapt the Self-Instruct (Wang et al., 2022) method by first generating constraints and then generating instructions. In that case, the decomposition step can be omitted.

3.2 Token-aware Preference Optimization

A well-known issue with DPO is its uniform treatment of all tokens in both chosen and rejected examples (Wu et al., 2023; Cao et al., 2024; Li et al., 2024). However, different tokens within responses carry varying significance. Especially in scenarios involving complex instructions, the responses tend to be lengthy and multi-facet. On one hand, not all tokens in the rejected response are erroneous and should be disapproved. On the other hand, chosen response may also contain tokens that fail to meet specific constraints, therefore should not be unanimously approved.

Despite previous researchers have explored fine-grained supervision signals, the signals either come from a stronger model (Cao et al., 2024; Li et al., 2024) or human annotation (Wu et al., 2023; Lightman et al., 2023). However, in our case, it is difficult for the model to provide accurate supervision for its own response, especially when dealing with multifaceted instructions and the evaluation is at token-level. Therefore, we propose Confidence-Guided Token-aware DPO, which obtains token-level supervision based on model confidence.

3.2.1 Preliminary: Token-level DPO

Direct Preference Optimization (DPO) (Rafailov et al., 2024b) proposes a direct optimization objective that satisfies the optimal preference policy without using a reward model:

$$\begin{aligned}&\mathcal{L}_{DPO}(\pi_\theta;\pi_{ref}) = \\ &- E_{(x,y^w,y^l)\sim\mathcal{D}}\left[\log\sigma\left(\beta\log\frac{\pi_\theta(y_w\mid x)}{\pi_{ref}(y_w\mid x)}\right.\right. \\ &\left.\left.-\beta\log\frac{\pi_\theta(y_l\mid x)}{\pi_{ref}(y_l\mid x)}\right)\right], \qquad (1)\end{aligned}$$

where π_θ and π_{ref} represent the policy model and the reference model, respectively.

Subsequently, based on the theories of Levine (2018), Rafailov et al. (2024a) derived the form of DPO in token-level Markov Decision Process[2], where dynamic weight can be easily integrated for different tokens[3], with the loss function as follows:

$$\begin{aligned}&\mathcal{L}_{TDPO}(\pi_\theta, D) = \\ &- \mathbb{E}_{(\tau_w,\tau_l)\sim D}\log\sigma(\beta\sum_{t=0}^{N-1} r_t^w\log\frac{\pi_\theta(\mathbf{a}_t^w|\mathbf{s}_t^w)}{\pi_{ref}(\mathbf{a}_t^w|\mathbf{s}_t^w)} \\ &-\beta\sum_{t=0}^{M-1} r_t^l\log\frac{\pi_\theta(\mathbf{a}_t^l|\mathbf{s}_t^l)}{\pi_{ref}(\mathbf{a}_t^l|\mathbf{s}_t^l)}), \qquad (2)\end{aligned}$$

where τ^w and τ^l represent the winning and losing trajectories, with N and M as the token numbers, and r_t represents the weight for the t-th token.

3.2.2 Calibrated Confidence as Token Weight

While Section 3.2.1 provide theoretical support for token-level DPO, it is non trivial to derive token-level supervision. In this work, we propose to use the calibrated confidence as supervision.

Given an instruction x, we obtain the entropy of probability distribution over target vocabulary of size V at each decoding step as the weights:

$$\text{Ent}(y_t|x^w,\theta) = -\sum_{v=1}^{V} p(y_t^v)\text{log}p(y_t^v), \qquad (3)$$

where $p(y_t)$ represents the conditional distribution $p(y_t|x, y_{<t}, \theta)$, and θ represents model parameters. If the majority of the probability mass is concentrated on a limited number of vocabulary words, it indicates that the model is confident and the token is more likely to be aligned with the instruction (Fomicheva et al., 2020). Conversely, if the probabilities resemble a uniform distribution, the resulting token is expected to be misaligned.

[2]Please refer to Appendix C.1 for more details.

[3]Please refer to Appendix C.2 for the mathematical proof of the dynamic token weight in preference optimization

While there are other attributes that could also impact the confidence score (such as fluency), in this work, we want to focus our optimization on instruction alignment. Therefore, we apply calibration to the entropy to derive the token-level supervision for chosen and rejected samples respectively:

$$r_t = \begin{cases} \text{Ent}(y_t|x^w,\theta)/\text{Ent}(y_t|x^l,\theta), & \text{for } y_t \text{ in } y^w, \\ \text{Ent}(y_t|x^l,\theta)/\text{Ent}(y_t|x^w,\theta), & \text{for } y_t \text{ in } y^l, \end{cases}$$

$$r_t = \mathbf{min}(\Gamma, r_t), \tag{4}$$

where the chosen sample (x^w, y^w) refers to $(\texttt{Ins}, \texttt{Resp})$, and the rejected sample (x^l, y^l) refers to $(\texttt{Ins}_{drop}, \texttt{Resp}_{drop})$, and Γ is an upper-bound to avoid extreme cases to disrupt training, and we set Γ as 2 in this work.

The rationale is straightforward: for a given token in the response, if it exhibits high confidence under x^w and low confidence under x^l, this suggests that the token aligns well with x^w but does not fit x^l, potentially reflecting the dropped constraint. Therefore, the token requires a larger weight if it is in the rejected response (or a smaller weight if it is in the chosen response).

Calibrated confidence guided token-aware DPO allows for more targeted optimization, focusing on tokens that highlight the constraint mismatch on complex instructions, instead of unanimously optimizing all tokens, thereby improving the efficiency of complex instruction alignment.

4 Experiments

4.1 Set-up

Models. We conduct experiments on two models: LLaMA-3-8B-Instruct (Dubey et al., 2024) and Qwen2-7B-Instruct (Yang et al., 2024a). Both models have undergone alignment to possess fundamental instruction-following ability.

Setting. The experiments are carried out in two distinct settings:

1. **Pre-Existing Instructions (PreInst):** We leverage pre-existing complex instructions as a starting point for the model. We randomly select 2,000 instructions from the dataset of WizardLM (Xu et al., 2024).
2. **Self-Generated Instructions (SelfInst):** In this setting, we generate instructions using the Self-Instruct method (Wang et al., 2022), based on 10 high-quality samples from Qin et al. (2024) as in-context examples. Compared with **PreInst**, this setting is more challenging as we need to construct the complex instructions from scratch.

Evaluation. We mainly perform evaluations on three complex instruction-following benchmarks: **CFBench** (Zhang et al., 2024b), **FollowBench** (Jiang et al., 2024b) and **ComplexBench** (Wen et al., 2024). We also conduct evaluations on one general instruction benchmark: **AlpacaEval2** (Dubois et al., 2024). Note that all benchmarks require GPT-4 for judgment, and we use GPT-4o-0513[4] as the evaluator for all of them.

Baselines. We mainly compared our method against the following self-alignment methods:

- **Self-Reward** (Yuan et al., 2024): This method leverages the model to first generate multiple responses and then construct rewards.
- **Self-Reward + BSM**: Based on Self-Reward, this method performs fine-grained evaluation based on BSM (Saha et al., 2024).
- **Self-Correct** (Palmeira Ferraz et al., 2024): This method generates initial output and then corrects it to construct preference data.
- **ISHEEP** (Liang et al., 2024): This method self-creates additional instruction-output pair, which are filtered for supervised fine-tuning.

4.2 Main Results

As demonstrated in Table 1, our proposed MuSC achieves significant improvement across both complex and general instruction-following benchmarks. The improvement is consistent among different settings, verifying its scalability. By creating preference data with both constraint-aware and token-aware contrast, the model effectively learns to address all constraints lying in the instructions.

The results of Self-Reward underperform our method, even with the help of branched evaluation (Saha et al., 2024). This is because of the limited evaluation capability of the model, especially when evaluating its own response to complex instructions. Moreover, as different responses generated from the same model to the same instruction typically do not vary significantly, it is difficult to create effective contrast samples with real negativity.

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

Setting	Method	CF-Bench CSR	CF-Bench ISR	CF-Bench PSR	FollowBench HSR	FollowBench SSR	ComplexBench Overall	AlpacaEval2 LC (%)	AlpacaEval2 Avg. Len
Results on LLaMA-3-8B-Instruct									
LLaMA-3-8B-Instruct		0.64	0.24	0.34	62.39	73.07	61.49	21.07	1702
SelfInst	Self-Reward	0.65	0.26	0.35	61.20	72.22	62.45	19.21	1824
	Self-Reward w/ BSM	0.68	0.28	0.39	64.30	73.84	64.13	19.03	1787
	Self-Reward w/ GPT-4	0.66	0.25	0.37	62.18	73.34	64.05	19.55	1767
	Self-Correct	0.52	0.20	0.27	54.38	67.19	55.91	7.97	1919
	ISHEEP	0.60	0.29	0.40	62.77	72.86	62.67	22.00	1707
	MuSC	**0.70**	**0.32**	**0.44**	**66.71**	**74.84**	**65.98**	**23.87**	1708
PreInst	Self-Reward	0.66	0.27	0.37	60.88	72.17	62.03	19.93	1789
	Self-Reward w/ BSM	0.68	0.29	0.40	63.96	73.78	64.3	20.98	1829
	Self-Reward w/ GPT-4	0.66	0.26	0.37	64.02	73.26	63.52	18.02	1804
	Self-Correct	0.60	0.23	0.32	60.11	70.94	60.79	6.20	1593
	ISHEEP	0.67	0.29	0.40	63.54	73.21	62.92	20.23	1703
	SFT	0.56	0.20	0.26	50.06	66.48	53.93	10.00	1079
	MuSC	**0.69**	**0.30**	**0.42**	**66.90**	**75.11**	**64.73**	**23.74**	1631
Results on Qwen2-7B-Instruct									
Qwen2-7B-Instruct		0.74	0.36	0.49	59.81	71.69	67.24	15.53	1688
SelfInst	Self-Reward	0.75	0.38	0.50	55.36	69.71	66.98	16.81	1756
	Self-Reward w/ BSM	0.75	0.38	0.50	57.83	70.53	67.02	16.94	1710
	Self-Correct	0.67	0.28	0.38	51.98	67.89	64.41	14.01	1497
	ISHEEP	0.76	0.40	0.52	57.01	69.88	67.32	16.99	1619
	MuSC	**0.78**	**0.42**	**0.54**	**62.60**	**72.57**	**69.39**	**20.08**	1595
PreInst	Self-Reward	0.75	0.37	0.49	56.45	70.00	66.45	15.98	1796
	Self-Reward w/ BSM	0.75	0.37	0.49	58.02	70.62	67.43	17.17	1764
	Self-Correct	0.66	0.28	0.37	49.47	66.35	64.32	14.46	1737
	ISHEEP	0.77	0.41	0.52	55.52	69.62	67.13	16.52	1627
	SFT	0.72	0.35	0.46	47.36	64.67	65.89	9.52	979
	MuSC	**0.79**	**0.44**	**0.55**	**62.73**	**73.09**	**70.00**	**20.29**	1613

Table 1: Experiment results of different groups of methods on instruction following benchmarks. For more detailed results on each benchmark, please refer to Appendix D.

The improvement of I-SHEEP also underperforms, likely due to its reliance on supervised fine-tuning for optimization. Previous research also suggests that learning from negative samples is more effective than learning solely from positive ones (Yang et al., 2024b). The results of Self-Correct degrades a large margin, which might be due to the inability of the model for self-correction on complex instructions (Palmeira Ferraz et al., 2024).

On general instruction benchmarks, our method also achieves significant improvement. This aligns with the previous research, which suggests that the improvement on complex instruction-following is beneficial for the overall instruction-following ability (Xu et al., 2024; Elmadany et al., 2023).

5 Analysis

5.1 MuSC on SFT model

In Section 4, our main experiments are conducted on the Instruction-version models. To exclude the influence of an initial preference optimization process, we apply our method on SFT models.

Specifically, we selected two SFT-versions of LLaMA models, LLaMA-3-8B-UltraChat-200K and LLaMA-3-8B-Tulu-330K[5]. Both models have gone through and only through SFT process on open-sourced datasets. As shown in Table 2, our proposed MuSC can improve both the complex and general instruction-following ability of SFT models by a large margin. Notice we only apply 2K samples when performing preference optimization, which is roughly 1% of the amount of SFT data.

[5]https://huggingface.co/Magpie-Align

Setting	Method	CF-Bench			FollowBench		ComplexBench	AlpacaEval2	
		CSR	ISR	PSR	HSR	SSR	Overall	LC (%)	Avg. Len
LLaMA-3-8B-Ultrachat-200K		0.54	0.18	0.25	33.64	51.37	44.89	5.94	861
PreInst	Self-Reward w/BSM	0.58	0.2	0.28	39.24	56.82	53.69	9.64	1099
	Self-Correct	0.32	0.08	0.01	18.59	36.88	36.38	3.47	575
	ISHEEP	0.57	0.19	0.26	42.22	58.87	52.95	10.35	1283
	MuSC	**0.66**	**0.25**	**0.34**	**48.93**	**61.54**	**56.24**	**11.13**	1112
LLaMA-3-8B-Tulu-330K		0.56	0.20	0.27	36.26	54.66	54.52	6.00	992
PreInst	Self-Reward w/BSM	0.60	0.22	0.30	41.71	58.68	55.84	9.71	1307
	Self-Correct	0.31	0.08	0.10	24.67	39.94	40.44	3.21	623
	ISHEEP	0.61	0.22	0.30	43.16	59.21	58.20	10.77	1402
	MuSC	**0.70**	**0.28**	**0.40**	**51.26**	**63.94**	**62.98**	**13.20**	1341

Table 2: Experiment results of different methods on supervised fine-tuned models.

Method	CF-Bench			AlpacaEval2	
	CSR	ISR	PSR	LC	Len
Baseline	0.64	0.24	0.34	21.07	1702
Perplexity	0.70	0.32	0.43	22.99	1744
PMI	0.69	0.29	0.41	21.92	1713
KLDiv	0.69	0.31	0.42	21.86	1686
Entropy	**0.71**	**0.34**	**0.44**	**23.74**	1631
w/o calib	0.68	0.28	0.39	21.49	1735

Table 3: Results of different confidence metrics as the fine-grained weight on LLaMA-3-8B-Instruct.

Method	CF-Bench			AlpacaEval2	
	CSR	ISR	PSR	LC	Len
Baseline	0.64	0.24	0.34	21.07	1702
Results on SelfInst					
DPO$_{MuSC}$	**0.70**	**0.32**	**0.44**	**23.87**	**1708**
w/o fgct	0.68	0.28	0.39	21.49	1735
SimPO$_{MuSC}$	**0.70**	**0.31**	**0.42**	**22.92**	**1716**
w/o fgct	0.67	0.30	0.40	19.59	1637
IPO$_{MuSC}$	**0.73**	**0.37**	**0.48**	**23.18**	**1650**
w/o fgct	0.70	0.33	0.44	20.42	1686
Results on PreInst					
DPO$_{MuSC}$	**0.69**	**0.30**	**0.42**	**23.74**	**1631**
w/o fgct	0.69	0.30	0.41	20.96	1671
SimPO$_{MuSC}$	**0.69**	**0.31**	**0.42**	**23.28**	**1625**
w/o fgct	0.67	0.31	0.41	22.02	1570
IPO$_{MuSC}$	**0.68**	**0.32**	**0.44**	**24.56**	**1601**
w/o fgct	0.68	0.30	0.42	22.04	1531

Table 4: Results of our method on different preference optimization methods on Llama-3-8B-Instruct.

This again verifies that learning from negative samples is comparatively more efficient than learning solely from positive samples.

5.2 The Influence of Confidence Metrics

Various confidence metrics have been established in the domain of LLM (Geng et al., 2024). This section aims to provide a comparison across different metrics as the token weight, under the framework of MuSC. We include the following metrics:

- **Perplexity**: The exponential of the negative log-likelihood of the token.
- **PMI**: Pointwise Mutual Information as defined in Takayama and Arase (2019).
- **KlDiv**: Kullback–Leibler divergence between the token probability distribution under chosen and rejected instructions.

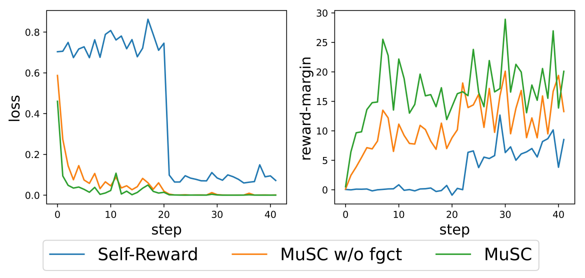
As shown in Table 3, entropy-based token weight achieves the best result among all metrics, verifying its effectiveness. Both the perplexity and PMI-based score underperforms, as they only consider the probabilities of the selected tokens instead of the whole distribution, leading to biased evaluation[6]. KLDiv-based score also underperforms, this is because KLDiv is essentially a distance measurement instead of a confidence measurement, which is not adapted to our scenario.

We also experiment with removing the calibration proposed in Section 3.2.2. As can be seen, calibration is important for the effectiveness of confidence-based fine-grained weight, as it can exclude other factors such as fluency, thereby focus-

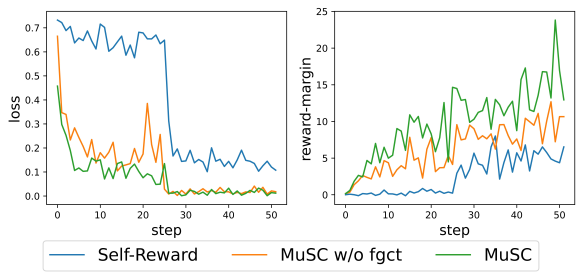
[6]As the model would always select (one of) the tokens with the highest probability at each step, relying only on the the selected tokens for evaluation will result in overconfidence.

Method	MMLU	GSM8K	HumanEval	Avg.
Results on Meta-LLaMA-3-8B-Instruct				
Baseline	68.29	79.08	59.15	51.63
MuSC$_{SelfInst}$	68.24	**79.23**	**62.20**	**52.42**
w/o fgct	67.97	78.62	56.71	50.83
MuSC$_{PreInst}$	68.17	77.41	**62.80**	**52.10**
w/o fgct	67.80	77.71	**60.98**	51.62
Results on Qwen2-7B-Instruct				
Baseline	70.76	83.09	75.61	57.37
MuSC$_{SelfInst}$	70.63	**84.09**	75.61	**57.58**
w/o fgct	70.81	82.94	71.34	56.27
MuSC$_{PreInst}$	70.46	**84.38**	75.78	**57.66**
w/o fgct	70.63	**84.53**	73.78	57.24

Table 5: Experiment results of our methods on fundamental capability benchmarks. Bolded results denote improvements, while grayed results denote degradation.



(a) Experiment Results on Llama3-8B-Instruct.



(b) Experiment Results on Qwen2-7B-Instruct.

Figure 3: Different statistical indicators during the training steps, upon different methods.

ing the contrast on instruction alignment.

5.3 Can MuSC Scale to Other xPO Method?

While our experiments primarily focus on the DPO method, the overall framework is not limited to a single preference optimization technique. Therefore, we extended our framework to two additional xPO methods: SimPO (Meng et al., 2024) and IPO (Azar et al., 2024). We utilized the same constructed preference data and applied the entropy-based score as the token-level supervision[7]. Note that for "w/o fgct", we just remove the fine-grained contrast in the MuSC method. In Table 4, our approach has shown consistent improvements across both SimPO and IPO, validating its scalability.

While IPO showed the best performance among different xPO methods, these performance differences were not statistically significant. Our method is not limited to DPO; instead, it's a general framework applicable to most xPO-style methods. Given the popularity of DPO, we primarily report our experiments based on DPO in this work.

5.4 Is Fundamental Capability Harmed?

Previous research have proposed that during the alignment process, the fundamental ability of model may suffer degradation due to alignment tax (Ouyang et al., 2022). Therefore, we evaluated our proposed method on three fundamental capability benchmarks: MMLU (Hendrycks et al., 2021), GSM8K (Cobbe et al., 2021) and HumanEval (Chen et al., 2021).

[7]Please refer to Appendix A.2 for detailed implementation.

As shown in Table 5, while the results of naive MuSC may suffer slight degradation, the introduction of fine-grained contrast mitigates the degradation, which verifies the significance of token-level supervision. Under the scenario of complex instruction, the response is lengthy and should not be uniformly approved or disapproved. With fine-grained supervision, we focus the optimization on complex instruction alignment, thereby avoiding the disruption of other capabilities.

5.5 The Variation of Statistical Indicators

As different methods start from the same group of instructions, training statistics can be comparable as a quality indicator. Therefore, we display the variation of both loss and reward margins between chosen and rejected samples on different methods.

As shown in Figure 3, Self-Reward presents both higher loss and lower reward margin during training. This is because there is too much noise between the chosen and rejected pairs, making the model unable to capture the contrast related to constraint alignment. Notice that both indicators start to change drastically at the 2nd epoch, which means the learned knowledge cannot transfer between different samples at the 1st epoch. On the other hand, the optimization based on MuSC converges faster and more smoothly, verifying the effectiveness of the contrast samples.

Comparing the indicators of MuSC with and

Model	CFBench			AlpacaEval2	
	CSR	ISR	PSR	LC	Len
Qwen2.5-1.5B-Inst	0.59	0.21	0.28	10.10	1507
+MuSC	**0.69**	**0.32**	**0.41**	**11.44**	1346
Qwen2.5-3B-Inst	0.72	0.34	0.45	27.44	2188
+MuSC	**0.76**	**0.41**	**0.52**	**30.64**	1873
Qwen2.5-14B-Inst	0.82	0.49	0.60	46.05	1792
+MuSC	**0.85**	**0.56**	**0.67**	**47.61**	1770
Qwen2.5-32B-Inst	0.87	0.60	0.70	48.63	1785
+MuSC	**0.89**	**0.65**	**0.75**	**49.34**	1772

Table 6: Experiment results of MuSC on Qwen2.5-Instruct of different model sizes.

without fine-grained supervision, it can also be noticed that with the introduction of fine-grained supervision, both indicators converge faster. The introduction of token-level supervision is a cheap yet effective method to improve xPO methods.

For the analysis of different instruction noising schemes and the visualization of token-level weight, please refer to Appendix B and D.

5.6 Can MuSC Scale among Model Sizes

To verify the scalability of our proposed method on models with larger sizes, we conducted extensive experiments on Qwen-2.5-Instruct models ranging from 1.5B to 32B parameters. As shown in Table 6, the results demonstrate consistent improvements across both smaller and larger models, confirming that our method is effective regardless of model size. This robust scalability is attributed to our approach being inherently self-improving and not requiring external models, which holds significant promise for advancing LLM capabilities.

6 Conclusion

In this work, we propose a Multi-granularity Contrastive Training framework, to perform complex instruction alignment without the introduction of external supervision. Experiment results show our method achieves significant improvement on instruction alignment benchmarks, surpassing previous self-improvement methods by a large margin. In the future, we will apply our MuSC on the improvement of other capabilities, such as long-form generation, multi-modal generation, etc.

Limitations

Our work still has some limitations: 1) Due to time and resource limitation, we did not validate our method on larger models, such as LLaMA-70B. Validation on larger models could help to improve the credibility of our method. 2) We mainly relied on GPT-4 based LLM-as-a-Judge to evaluate the results. Despite it has been verified that GPT-4 based evaluation achieves high correlation with human evaluators (Zheng et al., 2023), incorporating human evaluation would further improve the credibility of our methods. 3) We did not scale our method to PPO-based optimization methods, which are also wildly used in recent alignment practice. The application of our method on traditional RL methods could further improve its utility.

Ethical Considerations

Since the aligned data is generated based on the Instruct version model, the chances of generating toxic content are low. To further validate this, we randomly selected 200 samples in total—100 samples for Llama-3-8B-Instruct and 100 samples for Qwen-2-7B-Instruct—and employed three graduate students to annotate them. The results showed no instances of toxic instructions or responses.

However, there is potential risk of knowledge forgetting related to safety alignment. To address this, we recommend incorporating additional safe alignment samples during MuSC training to reinforce the model's safety-related knowledge. We will include these recommendations in the ethical considerations section of the revised manuscript.

Acknowledgements

This work is supported by National Natural Science Foundation of China (62276077, 62376075, 62376076), Jiangsu Science and Technology Major Project (BG2024031) and Nanjing University AI & AI for Science Funding (2024300540).

References

Mohammad Gheshlaghi Azar, Zhaohan Daniel Guo, Bilal Piot, Remi Munos, Mark Rowland, Michal Valko, and Daniele Calandriello. 2024. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pages 4447–4455. PMLR.

Meng Cao, Lei Shu, Lei Yu, Yun Zhu, Nevan Wichers, Yinxiao Liu, and Lei Meng. 2024. Beyond sparse rewards: Enhancing reinforcement learning with language model critique in text generation. *Preprint*, arXiv:2401.07382.

Changyu Chen, Zichen Liu, Chao Du, Tianyu Pang, Qian Liu, Arunesh Sinha, Pradeep Varakantham, and Min Lin. 2024a. Bootstrapping language models with dpo implicit rewards. *arXiv preprint arXiv:2406.09760*.

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.

Yongrui Chen, Haiyun Jiang, Xinting Huang, Shuming Shi, and Guilin Qi. 2024b. Dog-instruct: Towards premium instruction-tuning data via text-grounded instruction wrapping. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 4125–4135.

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

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

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

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

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*.

AbdelRahim Elmadany, ElMoatez Billah Nagoudi, and Muhammad Abdul-Mageed. 2023. ORCA: A challenging benchmark for Arabic language understanding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9559–9586, Toronto, Canada. Association for Computational Linguistics.

Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. 2020. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555.

Jiahui Geng, Fengyu Cai, Yuxia Wang, Heinz Koeppl, Preslav Nakov, and Iryna Gurevych. 2024. A survey of confidence estimation and calibration in large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6577–6595, Mexico City, Mexico. Association for Computational Linguistics.

Shangmin Guo, Biao Zhang, Tianlin Liu, Tianqi Liu, Misha Khalman, Felipe Llinares, Alexandre Rame, Thomas Mesnard, Yao Zhao, Bilal Piot, Johan Ferret, and Mathieu Blondel. 2024. Direct language model alignment from online ai feedback. *Preprint*, arXiv:2402.04792.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024a. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *Preprint*, arXiv:2404.15846.

Qianyu He, Jie Zeng, Wenhao Huang, Lina Chen, Jin Xiao, Qianxi He, Xunzhe Zhou, Jiaqing Liang, and Yanghua Xiao. 2024b. 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, et al. 2024c. Chinese simpleqa: A chinese factuality evaluation for large language models. *arXiv 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*.

Hamish Ivison, Yizhong Wang, Jiacheng Liu, Zeqiu Wu, Valentina Pyatkin, Nathan Lambert, Noah A. Smith, Yejin Choi, and Hannaneh Hajishirzi. 2024. Unpacking dpo and ppo: Disentangling best practices for learning from preference feedback. *Preprint*, arXiv:2406.09279.

Yuxin Jiang, Bo Huang, Yufei Wang, Xingshan Zeng, Liangyou Li, Yasheng Wang, Xin Jiang, Lifeng Shang, Ruiming Tang, and Wei Wang. 2024a. Bridging and modeling correlations in pairwise data for direct preference optimization. *Preprint*, arXiv:2408.07471.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024b. 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. Association for Computational Linguistics.

Siyu Lai, Hui Huang, Dong Jing, Yufeng Chen, Jinan Xu, and Jian Liu. 2021. Saliency-based multi-view mixed language training for zero-shot cross-lingual classification. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 599–610, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Sergey Levine. 2018. Reinforcement learning and control as probabilistic inference: Tutorial and review. *ArXiv*, abs/1805.00909.

Shilong Li, Yancheng He, Hui Huang, Xingyuan Bu, Jiaheng Liu, Hangyu Guo, Weixun Wang, Jihao Gu, Wenbo Su, and Bo Zheng. 2024. 2d-dpo: Scaling direct preference optimization with 2-dimensional supervision. *Preprint*, arXiv:2410.19720.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". 2023. Openorca: An open dataset of gpt augmented flan reasoning traces. `https://https://huggingface.co/Open-Orca/OpenOrca`.

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. *Preprint*, arXiv:2408.08072.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step. *arXiv preprint arXiv:2305.20050*.

Aiwei Liu, Haoping Bai, Zhiyun Lu, Xiang Kong, Xiaoming Wang, Jiulong Shan, Meng Cao, and Lijie Wen. 2024. Direct large language model alignment through self-rewarding contrastive prompt distillation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9688–9712, Bangkok, Thailand. Association for Computational Linguistics.

Jiaheng Liu, Dawei Zhu, Zhiqi Bai, Yancheng He, Huanxuan Liao, Haoran Que, Zekun Wang, Chenchen Zhang, Ge Zhang, Jiebin Zhang, et al. 2025. A comprehensive survey on long context language modeling. *arXiv preprint arXiv:2503.17407*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *Preprint*, arXiv:2301.13688.

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*.

Renze Lou, Kai Zhang, and Wenpeng Yin. 2024. Large language model instruction following: A survey of progresses and challenges. *Computational Linguistics*, pages 1–10.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. In *Advances in Neural Information Processing Systems (NeurIPS)*.

OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal

Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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

Thomas Palmeira Ferraz, Kartik Mehta, Yu-Hsiang Lin, Haw-Shiuan Chang, Shereen Oraby, Sijia Liu, Vivek Subramanian, Tagyoung Chung, Mohit Bansal, and Nanyun Peng. 2024. LLM self-correction with DeCRIM: Decompose, critique, and refine for enhanced following of instructions with multiple constraints. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7773–7812, Miami, Florida, USA. Association for Computational Linguistics.

Xianghe Pang, Shuo Tang, Rui Ye, Yuxin Xiong, Bolun Zhang, Yanfeng Wang, and Siheng Chen. 2024. Self-alignment of large language models via monopolylogue-based social scene simulation. *arXiv preprint arXiv:2402.05699*.

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

Rafael Rafailov, Joey Hejna, Ryan Park, and Chelsea Finn. 2024a. From r to q^*: Your language model is secretly a q-function. *Preprint*, arXiv:2404.12358.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2024b. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

Swarnadeep Saha, Omer Levy, Asli Celikyilmaz, Mohit Bansal, Jason Weston, and Xian Li. 2024. Branch-solve-merge improves large language model evaluation and generation. *Preprint*, arXiv:2310.15123.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *Preprint*, arXiv:1707.06347.

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

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

Junya Takayama and Yuki Arase. 2019. Relevant and informative response generation using pointwise mutual information. In *Proceedings of the First Workshop on NLP for Conversational AI*, pages 133–138, Florence, Italy. Association for Computational Linguistics.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`.

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

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

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Zixu, Zhu, Xiang-Bo Mao, Sitaram Asur, Na, and Cheng. 2024. A comprehensive survey of llm alignment techniques: Rlhf, rlaif, ppo, dpo and more. *Preprint*, arXiv:2407.16216.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking complex instruction-following with multiple constraints composition. *Preprint*, arXiv:2407.03978.

Siwei Wu, Zhongyuan Peng, Xinrun Du, Tuney Zheng, Minghao Liu, Jialong Wu, Jiachen Ma, Yizhi Li, Jian Yang, Wangchunshu Zhou, et al. 2024a. A comparative study on reasoning patterns of openai's o1 model. *arXiv preprint arXiv:2410.13639*.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024b. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *Preprint*, arXiv:2407.19594.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023. Fine-grained human feedback gives better rewards for language model training. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024a. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*.

Zhen Yang, Ming Ding, Tinglin Huang, Yukuo Cen, Junshuai Song, Bin Xu, Yuxiao Dong, and Jie Tang. 2024b. Does negative sampling matter? a review with insights into its theory and applications. *Preprint*, arXiv:2402.17238.

Da Yin, Xiao Liu, Fan Yin, Ming Zhong, Hritik Bansal, Jiawei Han, and Kai-Wei Chang. 2023. Dynosaur: A dynamic growth paradigm for instruction-tuning data curation. *arXiv preprint arXiv:2305.14327*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Alexander Zhang, Marcus Dong, Jiaheng Liu, Wei Zhang, Yejie Wang, Jian Yang, Ge Zhang, Tianyu Liu, Zhongyuan Peng, Yingshui Tan, et al. 2025. Codecriticbench: A holistic code critique benchmark for large language models. *arXiv preprint arXiv:2502.16614*.

Ruize Zhang, Zelai Xu, Chengdong Ma, Chao Yu, Wei-Wei Tu, Shiyu Huang, Deheng Ye, Wenbo Ding, Yaodong Yang, and Yu Wang. 2024a. A survey on self-play methods in reinforcement learning. *Preprint*, arXiv:2408.01072.

Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Tao Zhang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, Wentao Zhang, and Zenan Zhou. 2024b. Cfbench: A comprehensive constraints-following benchmark for llms. *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, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2024. A survey of large language models. *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, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

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

Brian D. Ziebart. 2010. Modeling purposeful adaptive behavior with the principle of maximum causal entropy.

A Implementation Details

A.1 Token-aware Preference Data Construction

For all models that used for preference data construction, we adopt the following prompts presented in Figure 5, 7, 6, 8, 9 and 8. We set the temperate as 0.5 for all steps to ensure diversity. To ensure the data quality, we filter instructions with less than three constraints and more than ten constraints. We also filter preference pairs with the same chosen and rejected responses.

For constraint dropout, we set the dropout ratio α to 0.3 to ensure that negative examples are sufficiently negative, meanwhile not deviate too much from the positive sample. We avoid dropout on the first constraint, as it often establishes the foundation for the task, and dropping the first one would make the recombined instruction overly biased.

A.2 Token-aware Preference Optimization

Our experiments are based on Llama-Factory (Zheng et al., 2024), and we trained all models on 8 A100-80GB SXM GPUs. The `per_device_train_batch_size` was set to 1, `gradient_accumulation_steps` to 8, leading to an overal batch size as 64, and we used bfloat16 precision. The learning rate is set as 1e-06 with cosine decay,and each model is trained with 2 epochs. We set β to 0.2 for all DPO-based experiments, β as 3.0 and γ as 1.0 for all SimPO-based experiments, β as 1.0 for all IPO-based methods referring to the settings of Meng et al. (2024). All of the final loss includes 0.1x of the SFT loss.

B The Influence of Noising Scheme

Previous work has proposed various noising strategies in contrastive training (Lai et al., 2021). While we leverage Constraint-Dropout for negative sample generation, to make a fair comparison with other strategies, we implement the following strategies: 1) Constraint-Negate: Leverage the model to generate an opposite constraint. 2) Constraint-Substitute: Substitute the constraint with an unrelated constraint.

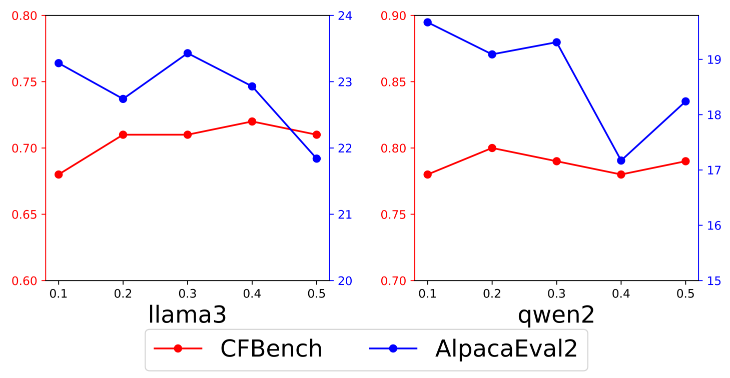


Figure 4: The variation of results on CFBench and AlpacaEval2 with different dropout ratios.

As shown in Table 7, both the negation and substitution applied on the constraints would lead to performance degradation. After a thoroughly inspect of the derived data, we realize that instructions derived from both dropout and negation would lead to instructions too far from the positive instruction, therefore the derived negative response would also deviate too much from the original instruction. An effective negative sample should fulfill both negativity, consistency and contrastiveness, and constrait-dropout is a simple yet effective method to achieve this goal.

We also provide the variation of the results on CF-Bench and AlpacaEval2 with different constraint dropout ratios. As shown in Figure 4, with the dropout ratio increased from 0.1 to 0.5, the results on CF-Bench firstly increases and then slightly decreases. On the other hand, the results on AlpacaEval2 declines a lot with a higher dropout ratio. This denotes that a suboptimal droout ratio is essential for the balance between complex instruction and general instruction following abilities, with lower ratio may decrease the effectiveness of general instruction alignment, while higher ratio may be harmful for complex instruction alignment. Finally, we set the constraint dropout ratio as 0.3 in all experiments.

Scenario	Method	Meta-LLaMA-3-8B-Instruct CF-Bench CSR	ISR	PSR	AlpacaEval2 LC%	Avg.Len	Qwen-2-7B-Instruct CF-Bench CSR	ISR	PSR	AlpacaEval2 LC%	Avg.Len
	baseline	0.64	0.24	0.34	21.07	1702	0.74	0.36	0.49	15.53	1688
PreInst	Constraint-Drop	**0.71**	**0.34**	**0.45**	**23.43**	1682	**0.79**	**0.43**	**0.54**	**19.31**	1675
	Constraint-Negate	0.68	0.28	0.39	18.94	1688	0.75	0.37	0.50	17.82	1663
	Constraint-Substitute	0.68	0.28	0.40	20.48	1706	0.76	0.39	0.51	19.05	1709

Table 7: Experiment results of different noising strategies on instruction following benchmarks.

C Mathematical Derivations

C.1 Preliminary: DPO in the Token Level Marcov Decision Process

As demonstrated in Rafailov et al. (2024a), the Bradley-Terry preference model in token-level Marcov Decision Process (MDP) is:

$$p^*\left(\tau^w \succeq \tau^l\right) = \frac{\exp\left(\sum_{i=1}^{N} r\left(\mathbf{s}_i^w, \mathbf{a}_i^w\right)\right)}{\exp\left(\sum_{i=1}^{N} r\left(\mathbf{s}_i^w, \mathbf{a}_i^w\right)\right) + \exp\left(\sum_{i=1}^{M} r\left(\mathbf{s}_i^l, \mathbf{a}_i^l\right)\right)} \tag{5}$$

The formula using the Q-function to measure the relationship between the current timestep and future returns:

$$Q^*(s_t, a_t) = \begin{cases} r(s_t, a_t) + \beta \log \pi_{ref}(a_t|s_t) + V^*(s_{t+1}), & \text{if } s_{t+1} \text{ is not terminal} \\ r(s_t, a_t) + \beta \log \pi_{ref}(a_t|s_t), & \text{if } s_{t+1} \text{ is terminal} \end{cases} \tag{6}$$

Derive the total reward obtained along the entire trajectory based on the above definitions:

$$\sum_{t=0}^{T-1} r(s_t, a_t) = \sum_{t=0}^{T-1} (Q^*(s_t, a_t) - \beta \log \pi_{\text{ref}}(a_t|s_t) - V^*(s_{t+1})) \tag{7}$$

Combining this with the fixed point solution of the optimal policy (Ziebart, 2010; Levine, 2018), we can further derive:

$$\sum_{t=0}^{T-1} r(s_t, a_t) = Q^*(s_0, a_0) - \beta \log \pi_{ref}(a_0|s_0) + \sum_{t=1}^{T-1} (Q^*(s_t, a_t) - V^*(s_t) - \beta \log \pi_{\text{ref}}(a_t|s_t)) \tag{8}$$

$$= Q^*(s_0, a_0) - \beta \log \pi_{ref}(a_0|s_0) + \sum_{t=1}^{T-1} \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \tag{9}$$

$$= V^*(s_0) + \sum_{t=0}^{T-1} \beta \log \frac{\pi^*(a_t|s_t)}{\pi_{\text{ref}}(a_t|s_t)} \tag{10}$$

By substituting the above result into Eq. 5, we can eliminate $V^*(S_0)$ in the same way as removing the partition function in DPO, obtaining the Token-level BT model that conforms to the MDP:

$$p_{\pi^*}\left(\tau^w \succeq \tau^l\right) = \sigma\left(\sum_{t=0}^{N-1} \beta \log \frac{\pi^*\left(\mathbf{a}_t^w \mid \mathbf{s}_t^w\right)}{\pi_{\text{ref}}\left(\mathbf{a}_t^w \mid \mathbf{s}_t^w\right)} - \sum_{t=0}^{M-1} \beta \log \frac{\pi^*\left(\mathbf{a}_t^l \mid \mathbf{s}_t^l\right)}{\pi_{\text{ref}}\left(\mathbf{a}_t^l \mid \mathbf{s}_t^l\right)}\right) \tag{11}$$

Thus, the Loss formulation of DPO at the Token level is:

$$\mathcal{L}\left(\pi_\theta, \mathcal{D}\right) = -\mathbb{E}_{(\tau_w, \tau_l) \sim \mathcal{D}}\left[\log \sigma\left(\left(\sum_{t=0}^{N-1} \beta \log \frac{\pi^*\left(\mathbf{a}_t^w \mid \mathbf{s}_t^w\right)}{\pi_{\text{ref}}\left(\mathbf{a}_t^w \mid \mathbf{s}_t^w\right)}\right) - \left(\sum_{t=0}^{M-1} \beta \log \frac{\pi^*\left(\mathbf{a}_t^l \mid \mathbf{s}_t^l\right)}{\pi_{\text{ref}}\left(\mathbf{a}_t^l \mid \mathbf{s}_t^l\right)}\right)\right)\right] \tag{12}$$

C.2 Proof of Dynamic Token Weight in Token-level DPO

In classic RLHF methods, the optimization objective is typically formulated with an entropy bonus, expressed through a Kullback-Leibler (KL) divergence constraint as follows:

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \sum_{t=0}^{T} [r(\mathbf{s}_t, \mathbf{a}_t) - \beta \mathcal{D}_{KL}[\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) || \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)]] \tag{13}$$

$$= \max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \sum_{t=0}^{T} [r(\mathbf{s}_t, \mathbf{a}_t) - \beta \log \frac{\pi_\theta(\mathbf{a}_t|\mathbf{s}_t)}{\pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)}] \tag{14}$$

This can be further rewritten by separating the terms involving the reference policy and the entropy of the current policy:

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} [\sum_{t=0}^{T} (r(\mathbf{s}_t, \mathbf{a}_t) + \beta \log \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)) + \beta \mathcal{H}(\pi_\theta) | \mathbf{s}_0 \sim \rho(\mathbf{s}_0)]$$

When the coefficient β is treated as a variable that depends on the timestep t (Li et al., 2024), the objective transforms to:

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \sum_{t=0}^{T} [(r(\mathbf{s}_t, \mathbf{a}_t) + \beta_t \log \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)) - \beta_t \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)] \tag{15}$$

where β_t depends solely on $\mathbf{a}_t$ and $\mathbf{s}_t$. Following the formulation by Levine (2018), the above expression can be recast to incorporate the KL divergence explicitly:

$$\max_{\pi_\theta} \mathbb{E}_{a_t \sim \pi_\theta(\cdot|\mathbf{s}_t)} \sum_{t=0}^{T} [(r(\mathbf{s}_t, \mathbf{a}_t) + \beta_t \log \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)) - \beta_t \log \pi_\theta(\mathbf{a}_t|\mathbf{s}_t)] \tag{16}$$

where the value function $V(\mathbf{s}_t)$ is defined as:

$$V(\mathbf{s}_t) = \beta_t \log \int_{\mathcal{A}} [\exp \frac{r(\mathbf{s}_t, \mathbf{a}_t)}{\beta_t} \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)] \, d\mathbf{a}_t \tag{17}$$

When the KL divergence term is minimized—implying that the two distributions are identical—the expectation in Eq. (14) reaches its maximum value. Therefore, the optimal policy satisfies:

$$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = \frac{1}{\exp(V(\mathbf{s}_t))} \exp\left(\frac{r(\mathbf{s}_t, \mathbf{a}_t) + \beta_t \log \pi_{ref}(\mathbf{a}_t|\mathbf{s}_t)}{\beta_t}\right) \tag{18}$$

Based on this relationship, we define the optimal Q-function as:

$$Q^*(s_t, a_t) = \begin{cases} r(s_t, a_t) + \beta_t \log \pi_{ref}(a_t|s_t) + V^*(s_{t+1}), & \text{if } s_{t+1} \text{ is not terminal} \\ r(s_t, a_t) + \beta_t \log \pi_{ref}(a_t|s_t), & \text{if } s_{t+1} \text{ is terminal} \end{cases} \tag{19}$$

Consequently, the optimal policy can be expressed as:

$$\pi_\theta(\mathbf{a}_t|\mathbf{s}_t) = e^{(Q(\mathbf{s}_t, \mathbf{a}_t) - V(\mathbf{s}_t))/\beta_t} \tag{20}$$

By taking the natural logarithm of both sides, we obtain a log-linear relationship for the optimal policy at the token level, which is expressed with the optimial Q-function:

$$\beta_t \log \pi_\theta(\mathbf{a}_t \mid \mathbf{s}_t) = Q_\theta(\mathbf{s}_t, \mathbf{a}_t) - V_\theta(\mathbf{s}_t) \tag{21}$$

This equation establishes a direct relationship between the scaled log-ratio of the optimal policy to the reference policy and the reward function $r(\mathbf{s}_t, \mathbf{a}_t)$:

$$\beta_t \log \frac{\pi^*(\mathbf{a}_t \mid \mathbf{s}_t)}{\pi_{\text{ref}}(\mathbf{a}_t \mid \mathbf{s}_t)} = r(\mathbf{s}_t, \mathbf{a}_t) + V^*(\mathbf{s}_{t+1}) - V^*(\mathbf{s}_t) \tag{22}$$

Furthermore, following the definition by Rafailov et al. (2024a)'s definition, two reward functions $r(\mathbf{s}_t, \mathbf{a}_t)$ and $r'(\mathbf{s}_t, \mathbf{a}_t)$ are considered equivalent if there exists a potential function $\Phi(\mathbf{s})$, such that:

$$r'(\mathbf{s}_t, \mathbf{a}_t) = r(\mathbf{s}_t, \mathbf{a}_t) + \Phi(\mathbf{s}_{t+1}) - \Phi(\mathbf{s}_t) \tag{23}$$

This equivalence implies that the optimal advantage function remains invariant under such transformations of the reward function. Consequently, we derive why the coefficient $beta$ in direct preference optimization can be variable, depending on the state and action, thereby allowing for more flexible and adaptive policy optimization in RLHF frameworks.

D Detailed Experiment Results

In this section, we presented detailed experiment results which are omitted in the main body of this paper due to space limitation. The detailed experiment results of different methods on ComplexBench, FollowBench and AlpacaEval2 are presented in Table 8, 10 and 9. The detailed results for the ablative studies of confidence metrics is presented in Table 11. The detailed results for the ablative studies of confidence metrics is presented in Table 7. We also present a case study in Table 12, which visualize the token-level weight derived from calibrated confidence score.

Scenario	**Method**	**ComplexBench**							
		Meta-Llama3-8B-Instruct				**Qwen2-7B-Instruct**			
		Overall	**And**	**Chain**	**Selection**	**Overall**	**And**	**Chain**	**Selection**
baseline		61.49	57.22	57.22	53.55	67.24	62.58	62.58	58.97
SelfInst	Self-Reward	62.45	58.23	58.23	54.07	66.98	63.02	63.02	57.75
	w/ BSM	64.13	58.01	58.01	56.62	67.02	62.37	62.37	57.85
	w/ GPT-4	64.05	59.44	59.44	54.78	—	—	—	—
	Self-Correct	55.91	49.85	49.85	46.91	64.41	59.59	59.59	55.04
	ISHEEP	62.67	57.79	57.79	54.63	67.32	61.95	61.95	59.64
	MuSC	**65.98**	**63.45**	**63.45**	**55.96**	**69.39**	**65.45**	**65.45**	**59.79**
PreInst	Self-Reward	62.03	56.94	56.94	53.09	66.45	61.37	61.37	57.64
	w/ BSM	64.30	57.58	57.58	56.47	67.43	62.95	62.95	58.41
	w/ GPT-4	63.52	59.08	59.08	53.91	—	—	—	—
	Self-Correct	60.79	55.65	55.65	52.02	64.32	60.16	60.16	54.63
	ISHEEP	62.92	56.37	56.37	54.83	67.13	64.45	64.45	57.54
	SFT	53.93	45.77	45.77	44.09	65.89	60.16	60.16	57.39
	MuSC	**64.73**	**59.23**	**59.23**	**55.91**	**70.00**	**66.88**	**66.88**	**61.38**

Table 8: Detailed experiment results of different methods on ComplexBench.

Scenario	Method	FollowBench					
		Meta-Llama3-8B-Instruct			Qwen2-7B-Instruct		
		HSR	SSR	CSL	HSR	SSR	CSL
baseline		62.39	73.07	2.76	59.81	71.69	2.46
SelfInst	Self-Reward	61.20	72.22	2.56	55.36	69.71	2.34
	w/ BSM	64.30	73.84	2.80	57.83	70.53	2.41
	w/ GPT-4	62.18	73.34	2.66	—	—	—
	Self-Correct	54.38	67.19	2.02	51.98	67.89	2.16
	ISHEEP	62.77	72.86	2.52	57.01	69.88	2.36
	MuSC	**66.71**	**74.84**	**2.92**	**62.60**	**72.57**	**2.82**
PreInst	Self-Reward	60.88	72.17	2.64	56.45	70.00	2.44
	w/ BSM	63.96	73.78	2.66	58.02	70.62	2.42
	w/ GPT-4	64.02	73.26	2.64	—	—	—
	Self-Correct	60.11	70.94	2.70	49.47	66.35	1.98
	ISHEEP	63.54	73.21	2.64	55.52	69.62	2.28
	SFT	50.06	66.48	2.04	47.36	64.67	1.96
	MuSC	**66.90**	**75.11**	**2.99**	**62.73**	**73.09**	**2.86**

Table 9: Detailed experiment results of different methods on FollowBench.

```
Please break down the instruction into multiple constraints and an input, following the provided
examples. Do not generate any other openings, closings or other uncessary explanations.
Examples:

**Instruction:**
{instruction_body}
**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}

**Instruction:**
{instruction_body}
**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}

...
Below is the instruction that needs to be broken down:
**Instruction:**
{instruction_body}
**Constraints:**
```

Figure 5: The prompt template used for instruction decomposition.

Scenario	Method	AlpacaEval2					
		Meta-Llama3-8B-Instruct			Qwen2-7B-Instruct		
		LC (%)	WR (%)	Avg. Len	LC (%)	WR (%)	Avg. Len
baseline		21.07	18.73	1702	15.53	13.70	1688
SelfInst	Self-Reward	19.21	19.18	1824	16.81	15.66	1756
	w/ BSM	19.03	18.34	1787	16.94	15.09	1710
	w/ GPT-4	19.55	18.53	1767	—	—	—
	Self-Correct	7.97	9.34	1919	14.01	10.92	1497
	ISHEEP	22.00	19.50	1707	16.99	14.04	1619
	MuSC	**23.87**	**20.91**	**1708**	**20.08**	**15.67**	**1595**
PreInst	Self-Reward	19.93	19.04	1789	15.98	15.62	1796
	w/ BSM	20.98	20.75	1829	17.17	16.21	1764
	w/ GPT-4	18.02	17.74	1804	—	—	—
	Self-Correct	6.20	5.81	1593	14.46	14.02	1737
	ISHEEP	20.23	17.86	1703	16.52	13.36	1627
	SFT	10.00	6.22	1079	9.52	5.25	979
	MuSC	**23.74**	**19.53**	**1631**	**20.29**	**15.91**	**1613**

Table 10: Detailed experiment results of different methods on AlpacaEval2.

Scenario	Method	Meta-Llama-3-8B-Instruct					Qwen-2-7B-Instruct				
		CF-Bench			AlpacaEval2		CF-Bench			AlpacaEval2	
		CSR	ISR	PSR	LC (%)	Avg. Len	CSR	ISR	PSR	LC (%)	Avg. Len
PreInst	Baseline	0.64	0.24	0.34	21.07	1702	0.74	0.36	0.49	15.53	1688
	w/ perplexity	0.70	0.32	0.43	22.99	1744	0.79	0.43	0.54	19.31	1675
	w/ PMI	0.69	0.29	0.41	21.92	1713	0.78	0.43	0.55	17.42	1651
	w/ KLDiv	0.69	0.31	0.42	21.86	1686	0.78	0.42	0.54	18.73	1614
	w/ entropy	**0.71**	**0.34**	**0.44**	**23.74**	1631	**0.80**	**0.46**	**0.58**	**20.38**	1683

Table 11: Detailed experiment results of different confidence metrics as the fine-grained weight.

Instruction 1: Identify the incorrect word in the given sentence: "The group acquired the skill to ski in a suitable manner." Rewrite the identified incorrect word correctly, ensuring the new sentence adheres to proper capitalization, punctuation, and grammar rules. Include at least one synonym of the incorrect word to test the model's understanding of context and vocabulary, and ensure the synonym is used appropriately in the context of the sentence.
Chosen Response 1: The group acquired the skill to ski in a proficient manner . The incorrect word is "suitable" . A synonym for "suitable" is "appropriate" , but it doesn't fit the context of the sentence . "Proficient" is a better synonym in this context as it means skilled or expert .
Rejected Response 1: The word "acquired" seems incorrect in the given sentence . A better word to use in this context would be "learned" . Rewritten sentence : "The group learned the skill to ski in a suitable manner ."
Instruction 2: Rewrite the given sentence in C# syntax, using an idiom or metaphor to express disbelief, and ensure the sentence maintains its original meaning while being grammatically correct in C#. The sentence to be rewritten is: "The idea that pigs can fly seems more plausible than his explanation for being late."
Chosen Response 2: csharp\n using System ;\n class Program\n {\n static void Main (string [] args \n) {\n Console.WriteLine ("His explanation for being late is so baffling that it's akin to saying pigs can fly , making it more believable in comparison .) ;\n }\n }
Rejected Response 2: Wow , it's truly hard to believe that pigs flying could be more plausible than my explanation for being late !

Table 12: Visualization of dynamic weights derived for chosen and rejected responses, based on our proposed calibrated entropy score. We select two samples from the datasets as an illustration.

```
Please combine the provided input and constraints into an instruction, following the provided examples.
Examples:

**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}
**Instruction:**
{instruction_body}

**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}
**Instruction:**
{instruction_body}

...
Below are the input and constraints for constructing your Instruction:
**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}
Please generate the instruction directly. Please incorporate the content of the input inside the
instruction if it is not None. Do not generate any other openings or closings.
```

Figure 6: The prompt template used for constraint recombination.

```
Please follow the examples to come up with one task with more than 5 constraints. Please list the
constraints first and then provide the task.
Examples:

**Task:**
{instruction_body}
**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}

**Task:**
{instruction_body}
**Constraints:**
{constraint_body1}
{constraint_body2}
{constraint_body3}

...
```

Figure 7: The prompt template used for self-instruct.

```
The following is a question used to apply constraint for generating an instruction. Please generate a
new constraint question which specifies a constraint on the contrary.
Please generate the constraint question with the following format: '**New Constraint**: [new constraint
here]. Do not generate any other openings, closings or explanations.
**Original Constraint**: {constraint_body}
```

Figure 8: The prompt template used for constraint substitution.

```
The following is a question used to apply constraint for generating an instruction. Please generate a
new constraint question which specifies a constraint deviates from the original one.
Please generate the constraint question with the following format: '**New Constraint**: [new constraint
here]. Do not generate any other openings, closings or explanations.
**Original Constraint**: {constraint_body}
```

Figure 9: The prompt template used for constraint negation.