Order Matters: Investigate the Position Bias in Multi-constraint Instruction Following

Anonymous ACL submission

Abstract

Real-world instructions with multiple constraints pose a significant challenge to existing large language models (LLMs). An observation is that the LLMs exhibit dramatic performance fluctuation when disturbing the order of the incorporated constraints. Yet, none of the existing works has systematically investigated this position bias problem in the field of multi-constraint instruction following. To bridge this gap, we design a probing task where we quantitatively measure the difficulty distribution of the constraints by a novel Difficulty Distribution Index (CDDI). Through the experimental results, we find that LLMs are more performant when presented with the constraints in a "hard-to-easy" order. This preference can be generalized to LLMs with different architecture or different sizes of parameters. Additionally, we conduct an explanation study, providing an intuitive insight into the correlation between the LLM's attention and constraint orders. Our code and dataset are publicly available at `https://anonymous.4open.science/r/woo-2009/`.

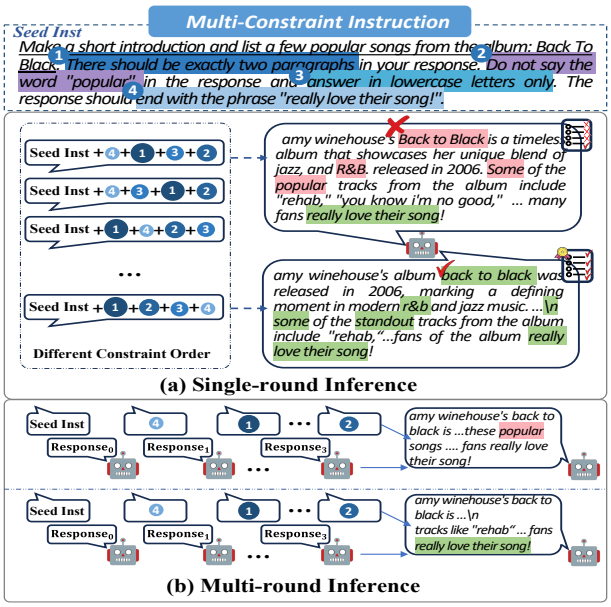


Figure 1: (a) In single-round inference, the LLM performs differently when handling the same instruction with different constraint orders. (b) In multi-round inference, the latter response is evitably affected by the former context.

1 Introduction

Large language models (LLMs) have made impressive progress in massive natural language tasks (Wan et al., 2024; Zhang et al., 2024b) and have been applied to various real-world scenarios (Bai et al., 2023; Bi et al., 2024). To achieve satisfactory performance, it is crucial for LLMs to understand the user's instructions and convey desired outputs, which is known as the Instruction Following capacity of LLM (Yin et al., 2023; Xu et al., 2024).

In practice, instructions are usually incorporated with multiple constraints of different types, e.g., format constraint which limits the model's output to a specific format. Nevertheless, existing LLMs often struggle to follow multi-constraint instructions, making multi-constraint instruction following an obstacle to hinder LLMs' real-world application (Wen et al., 2024; Yin et al., 2023).

Recently, a lot of works have demonstrated that LLMs are sensitive to the position of the referred context in many tasks, such as multi-document question answering, text evaluation, and list-wise ranking (Liu et al., 2024; Zheng et al., 2023; Tang et al., 2024). Since there are usually multiple constraints coexisting in the complex instruction, the position bias problem is also significant in multi-constraint instructions. As shown in Fig. 1, in the single-round scenario, the LLM's performance varies significantly when presented with instructions that have different constraint orders, even though the two instructions are semantically identical. When it comes to the multi-round scenario, different constraint orders impose different impacts

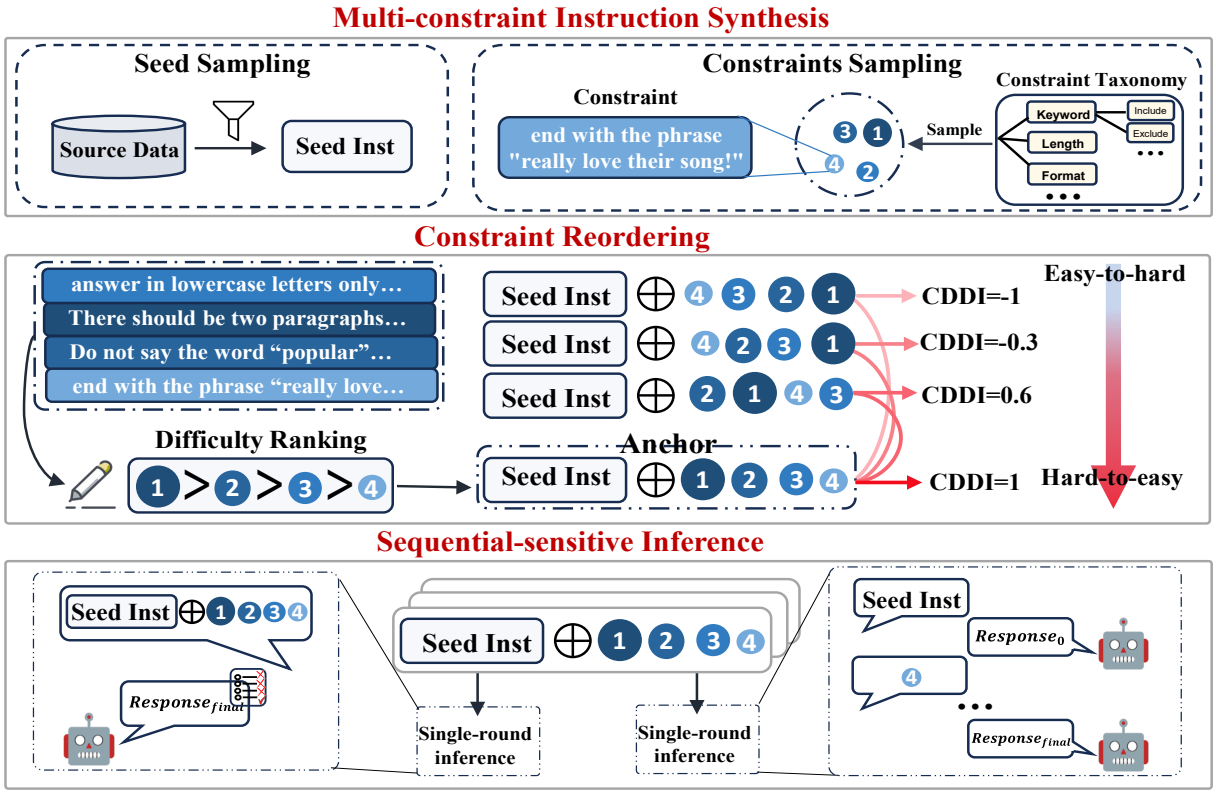


Figure 2: The procedure of the probing task. First, we synthesize the initial instructions by sampling seed instructions and corresponding constraints. Then, we obtain instructions with different constraint orders by reordering the incorporated constraints. Finally, we conduct model inference on single and multi-round settings.

on the intermediate responses, thus inevitably leading to a discrepancy in the quality of the final responses.

Nevertheless, the position bias of constraint orders in the multi-constraint instruction following remains an under-explored problem. Existing work manually assigns difficulty to different constraints based on a predefined rule and orders the constraints according to their difficulty. They empirically demonstrate the existence of LLMs' performance fluctuation brought by different constraint order (Chen et al., 2024). However, on the one hand, handcraft difficulty categorization fails to reflect the real difficulty disparity of different constraints (Dentella et al., 2024; Srivastava et al., 2023). On the other hand, they merely analyze the constraint order in a qualitative way, lacking a quantitative metric to measure the disparity of constraint order. Additionally, none of the existing works has provided an intuitive explanation for the position bias in multi-constraint instructions. It remains unclear how the LLMs handle instructions with different constraint orders.

To address all the problems above, we systematically investigate the position bias problem in the multi-constraint instructions. First, we propose a novel metric called the Constraint Difficulty Distribution Index (CDDI) to quantitatively describe the disparity of constraint order from the perspective of constraint difficulty. We leverage the accuracy of the LLM to quantify the difficulty of different constraints, thus precisely reflecting their disparity. Then, for a thorough study of the position bias problem, we design a probing task. As shown in Fig. 2, we construct a large number of multi-constraint instances with different constraint orders and explore two practical scenarios: single-round inference and multi-round inference. Our experiments find existing LLMs commonly perform better with the "hard-to-easy" constraint orders, i.e., possibly placing harder constraints in former positions. Finally, to make an intuitive explanation of our findings, we resort to a gradient-based method (Wu et al., 2023). We visualize the importance of different constraints located in different positions. We observe that the constraint order will affect how the LLM handle the constraints and is highly correlated to the LLM's performance on a specific constraint.

In summary, our main contributions are as follows: (1) We are the first to systematically investi-

gate the position bias problem in multi-constraint instruction following. (2) We propose a novel CDDI metric to quantify the disparity of different constraint orders in the multi-constraint instructions. (3) Through extensive experiments, we find that existing LLMs can achieve a better performance when presented with constraints in "hard-to-easy" orders. This finding can be generalized in both single-round and multi-round scenarios, regardless of the architecture of LLM, the size of LLM's parameters and the number of constraints. (4) Our explanation study explores how the LLMs assign attention when provided with instructions in different constraint orders and demonstrates the significant correlation between the attention patterns and the LLMs' performance on specific constraints.

2 Related Work

2.1 Complex Instruction Following

Riding on the wave of the large language model, the instruction following has attracted increasing attention for it is easy to be perceived by the users (Zhou et al., 2023a; Lou et al., 2024). Practical instructions are complex, usually incorporated with multiple constraints of different types (Zhou et al., 2023b; He et al., 2024). A lot of evaluation benchmarks have found that multi-constraint instruction following is nontrivial for the LLMs (Jiang et al., 2023b; Wen et al., 2024; Qin et al., 2024). Consequently, several works propose to improve the LLM's complex instruction following capacity by introducing additional instruction fine-tuning (Sun et al., 2024; Cheng et al., 2024; Zhang et al., 2024a).

Different from these works, we focus on the inference stage of the LLMs instead of model training. Especially, we aim to investigate the position bias problem brought by the constraint order, which poses an essential impact on the model performance.

2.2 Position Bias in the LLM

The position bias problem is common in the various LLM tasks (Liu et al., 2024; Zheng et al., 2023; Zeng et al., 2023). Researchers fisrt find that the LLM's performance degrades dramatically by merely changing the order of relevant information in the long-context question answering. A lot of works have studied the position bias problem in the field of logical reasoning (Chen et al.; Liu et al., 2023; Berglund et al., 2023). They find the LLM is sensitive to the order of premises, although such ordering actually does not alter the reasoning task (Chen et al.; Liu et al., 2023).

Despite so, none of these works has studied the position bias problem in the field of instruction following, especially multi-constraint instruction following. SIFo (Chen et al., 2024) is the most related work to ours. They manually differentiate the constraints based on the context length they will influence and conduct an empirical study to verify whether the model performance will be affected by the constraint order. However, Their investigation of position bias is fairly qualitative. Different from them, we are the first to make a systematical and thorough investigation on the position bias of constraints in multi-constraint instruction following.

3 Method

3.1 Background

In this paper, we mainly focus on the multi-constraint instruction I_c. It can be formulated as a seed instruction incorporated with n constraints:

$$I_c = I_s \oplus C_1 \oplus ... \oplus C_n, \quad (1)$$

where the seed instructions I_s describe a task, e.g., write a story, while these constraints $\sum_{i=1}^{n} C_i$ limit the output from different aspects, e.g., format, length, content, etc. $\oplus$ stands for the concatenation operation.

3.2 Probing Task

3.2.1 Task Formulation

To investigate the impact of constraint order, we introduce a probing task. In this task, the LLM is given multi-constraint instructions with constraints arranged in various orders. The LLM's task is to generate a response that follows all constraints. We evaluate the LLM in two practical scenarios: single-round and multi-round inference. The LLM's responses are then evaluated to determine its performance across various constraints. The overall procedure is illustrated in Fig. 2. In the following sections, we will provide a detailed explanation.

3.2.2 Multi-constraint Instruction Synthesis

To ensure the generalizability of probing data, we construct the initial multi-constraint instructions which include a variety of tasks and diverse constraint combinations. The multi-constraint instruction synthesis can be further divided into two parts: seed sampling and constraint sampling.

For the seed sampling, we sample data from three source datasets: (1) Natural Instructions V2 (Wang et al., 2022). It is an instruction collection covering more than 1600 NLP tasks. We filter those tasks that are too easy and could potentially conflict with complex constraints, e.g., object classification and sentiment tagging. Then, we randomly sample 52 instructions from the remaining tasks. (2) Self-Instruct (Wang et al., 2023). We only sample 83 instances from their initial 175 seed instructions which are formulated by humans. (3) Open Assistant (Köpf et al., 2024). Following the strategy of Suri (Li et al., 2023), we filter out the first turn of the conversation with the highest quality (marked as rank 0 in the dataset) and sample 65 instances from them. Overall, we obtain 200 seed instructions, where the number of instructions is denoted as n_{seed}.

As for the constraint sampling, we first categorize the constraints into 8 groups with 25 fine-grained types (Zhou et al., 2023a). For each type of constraint, we employ 8 different expressions to describe it[1]. Then, we sample n constraints from the constraint taxonomy and use the predefined rules to avoid possible conflicts. To ensure diversity, we repeat the sampling process to obtain n_{cc} distinct constraint combinations, deriving $n_{seed} \times n_{cc}$ multi-constraint instructions.

3.2.3 Constraint Reordering

To quantitatively construct instructions with different constraint orders, here are two questions that need to be answered: (1) *How do we distinguish the disparity of different constraints*? (2) After we order the constraints based on their disparity, *how do we quantitatively describe the disparity of constraint orders*?

An appropriate solution for the first question is to categorize the constraints based on their difficulty (Chen et al., 2024). In this paper, we also sort the constraints based on their difficulty. However, different from existing works which designate the difficulty of the constraints based on handcraft rules, we measure the difficulty of a constraint via the overall accuracy of following it in our probing datasets. The formulation is as follows:

$$\text{Dff}_{C_x} = \text{Softmax}(1 - \text{Acc}_{C_x}), \tag{2}$$

$$\text{Acc}_{C_x} = \frac{1}{N_x} \sum_{i=1}^{N_x} c_x^i. \tag{3}$$

[1]More details are shown in Appx. A.2

The C_x refers to a specific type of constraint, the N_x stands for the total number of instructions corresponding to the constraint C_x, and the c_x^i is a binary value to reflect whether the constraint C_x is followed in the ith instruction.

To quantitatively describe the disparity of constraint order, we propose a novel metric called the Constraint Difficulty Distribution Index (CDDI) which quantifies a specific constraint order based on its difficulty distribution. Given the difficulty of different types of constraints, we can readily attain the difficulty distribution of the constraints incorporated in the multi-constraint instructions. Specifically, for a multi-constraint instruction, we rank the incorporated constraints based on their difficulty, from the hardest to the easiest. We set this "hard-to-easy" constraint order as an anchor since it depicts an extreme situation, i.e., we designate the CDDI $= 1$ when the constraints fall in this order. Consequently, akin to the Kendall tau distance (Cicirello, 2020), we measure the difficulty distribution of a specific constraint order o by comparing it with the "hard-to-easy" constraint order o_{max}. The formula is shown as:

$$\text{CDDI}_o = \frac{N_{con} - N_{dis}}{N_{pair}} = \frac{2(N_{con} - N_{dis})}{n(n-1)}. \tag{4}$$

where N_{con} and N_{dis} represent the number of concordant and discordant distribution pairs of constraints between o and o_{max}, respectively. The N_{pair} is the total number of compared constraint pairs. Overall, we select n_{dd} different difficulty distributions, finally comprising $n_{seed} \times n_{cc} \times n_{dd}$ instances.

3.2.4 Sequential-Sensitive Inference

Given the multi-constraint instructions with different constraint orders, we evaluate the model's performance in two common scenarios: single-round inference and multi-round inference. In single-round inference, the LLM is directly given the multi-constraint instructions with different constraint distributions. We argue that different constraint distributions could impose different levels of difficulty on the LLM to handle. The multi-round inference introduces a more typical setting: the user will first provide the LLM with the core intention (i.e., the seed instruction in this work), and then iteratively put forward the constraints in order to obtain a final response.

To evaluate the model performance, apart from the constraint following accuracy mentioned in

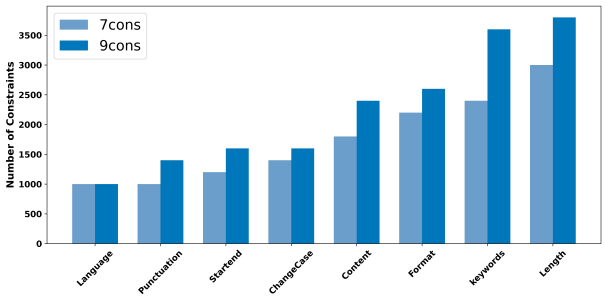


Figure 3: The statistic of different types of constraints in the probing data. The 7cons and 9cons stand for the setting when n=7 and n=9, respectively.

Eq.(3), we also verify its constraint-level accuracy Acc_{cons} and instruction-level accuracy Acc_{inst}. Corresponding formulas are shown below:

$$\text{Acc}_{\text{cons}} = \frac{1}{mn}\sum_{i=1}^{m}\sum_{j=1}^{n} c_i^j, \text{Acc}_{\text{inst}} = \frac{1}{m}\sum_{i=1}^{m}\prod_{j=1}^{n} c_i^j. \quad (5)$$

where m and n refer to the number of instructions and constraints in the instruction, respectively. Similar to Eq.(3), the c_i^j is a binary value which equals 1 when the constraint is followed in the ith instruction. All the evaluation is conducted by leveraging the script introduced in (Zhou et al., 2023a). We only evaluate the final responses produced by the LLMs.

4 Empirical Study

4.1 Experiment Setup

Models For our probing task, to ensure the generalizability of our study, we conduct experiments on both closed and open-source LLMs with varying architectures and parameter sizes. Specifically, we introduce the following models: (1) LLaMA3-8B-Instruct and LLaMA3-70B-Instruct (Dubey et al., 2024). (2) LLaMA2-13B-Chat (Touvron et al., 2023). (3) Mistral-7B-Instruct (Jiang et al., 2023a).[2] (4) Qwen2.5-7B-Instruct (Yang et al., 2024). (5) GPT4o-mini (Achiam et al., 2023).

Datasets We construct various multi-constraint instructions with different constraint orders (Sec.3.2). We empirically set the number of constraints n to 7. To ensure the diversity and complexity, we set the number of constraint combinations n_{cc} to 10 and the number of difficulty distributions n_{dd} to 12, finally obtaining $200 \times 10 \times 12 = 24$K samples. To verify the influence of constraint number, we also conduct experiments on the setting when $n = 9$. The statistic of the data for the probing task is provided in Fig. 3.

[2]We use the latest v0.3 version.

4.2 Results

LLMs prefer to "hard-to-easy" constraint distribution. As shown in Fig. 4, most of the LLMs exhibit a dramatic performance fluctuation on instructions with varying constraint distributions. When the constraint number is set to 7, the LLaMA3-8B-Instruct and Qwen2.5-7B-Instruct show approximately 7% and 5% performance disparity in extreme situations. This indicates the vulnerability of existing LLMs to the position bias brought by the constraint order. Also, the LLMs tend to be more performant to instructions with higher CDDI values. Even the LLaMA3-70B-Instruct exposes a clear preference for higher CDDI value as the number of constraints increases to 9, demonstrating that "hard-to-easy" is a superior constraint distribution for existing LLMs.

Multi-round inference exhibits more severe position bias compared with the single-round inference. The LLMs' performance in multi-round inference is presented in the Fig. 5. Compared with the results in the single-round inference, the performance gap becomes more prominent. All the LLMs gain approximately 10% improvement on C_level accuracy. Surprisingly, the LLaMA3-8B-Instruct and LLaMA3-70B-Instruct achieve approximately 25% performance improvement by changing the constraint distribution from "easy-to-hard" (CDDI=-1) to "hard-to-easy" (CDDI=1). This indicates that the LLMs are more sensitive to the position bias problem in a multi-round scenario.

LLMs perform better in multi-round inference when provided with the instructions in appropriate constraint order Comparing the results in single-round (Fig. 4) and multi-round inference (Fig. 5), we observe that the LLMs reach better performance if the incorporated constraints are arranged in an appropriate order. Specifically, when the CDDI value is negative, the performance of LLMs in multi-round inference lags behind that in single-round inference. Nevertheless, with the increase of the CDDI value, the LLMs can achieve superior performance in multi-round inference and reach their best performance in CDDI=1. An exception is the Mistral-7B-Instruct-v0.3. We attribute this to its inferiority in processing multi-round information (Chen et al., 2024).

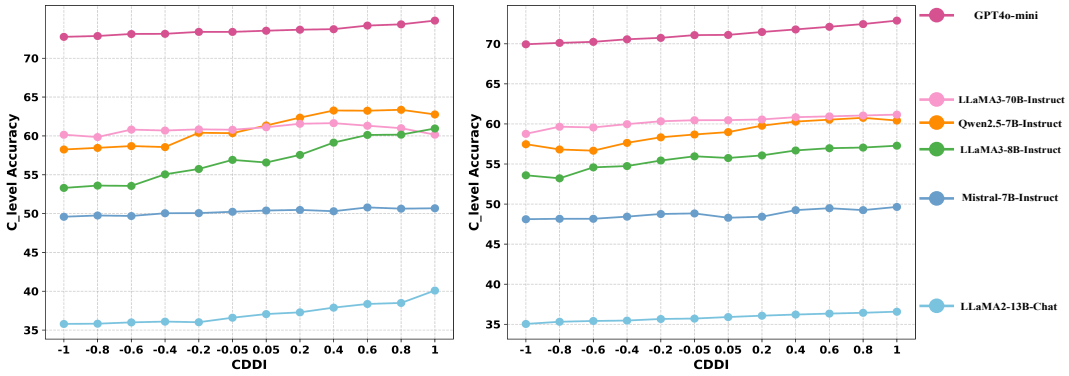


Figure 4: The performance of different LLMs in the single-round inference. The left and right figures show the results with the number of constraints n set to 7 and 9, respectively. With the increase of the CDDI, the constraint order changes from "easy-to-hard" to "hard-to-easy".

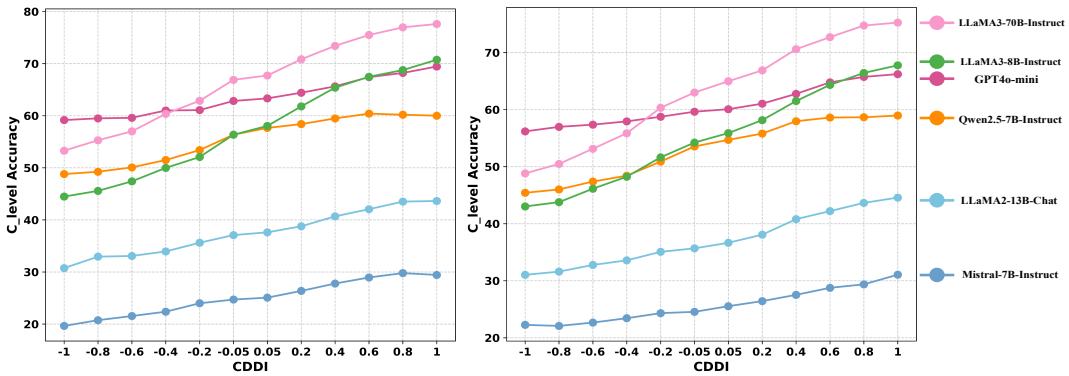


Figure 5: The performance of different LLMs in the multi-round inference. The left and right figures show the results with the number of constraints n set to 7 and 9, respectively. With the increase of the CDDI, the constraint order changes from "easy-to-hard" to "hard-to-easy".

Position bias varies in different types of constraints. We present the performance of the LLaMA3-8B-Instruct across different types of constraints in Tab. 1. As observed, with the increase of the CDDI value, the model's performance across most constraint types shows an upward trend except for Startend and Content, indicating that not all the constraints can benefit from the "hard-to-easy" constraint distribution in single-round inference. We make a more comprehensive explanation study in Sec. 5.3 for further investigation. Regarding the multi-round inference, the model's performance only exhibits a drop tendency in the Length type as the CDDI value increases, indicating that the LLMs struggle to generate a length-controlled final response when the length constraint is applied early in the multi-round inference (Yuan et al., 2024).

4.3 Robustness of CDDI

Since the CDDI is calculated by comparing the concordant and discordant pairs of two different constraint orders, there are usually multiple constraint orders sharing the same CDDI value. Therefore, we conduct a testing experiment to assess whether the LLM exhibits significant fluctuations across different constraint orders with the same CDDI value. Specifically, we set the CDDI to -0.05, a value that includes the most constraint orders in our setting, and conduct single-round inference for 3 times. The experiment results are shown in Tab 2. We calculate the P-value of the data, finding that the P-value is much larger than 0.05. This indicates that the fluctuation of LLM's performance is negligible among different constraint orders in the same CDDI value.

5 Explanation Study

5.1 Explanation Metric

To make an explanation for the influence brought by the constraints of different orders, we make an explanation study on where the LLMs mainly focus when handling multi-constraint instructions via a feature attribution-based explanation method (Li

CDDI	Length	Language	Punctuation	Format	Keywords	ChangeCase	Startend	Content	C_level	I_level
					Single-round Inference					
-1	27.50	28.20	23.30	71.14	68.58	49.57	62.92	**81.22**	53.30	1.95
-0.8	28.23	30.70	23.90	73.64	68.46	49.00	**63.50**	77.78	53.60	1.80
-0.6	28.53	31.10	26.60	71.23	69.25	49.79	60.92	78.22	53.56	1.95
-0.4	28.53	36.10	30.70	72.41	71.58	51.64	62.33	77.83	55.05	2.10
-0.2	29.33	39.30	35.30	73.82	72.08	50.07	60.75	77.44	55.74	2.40
-0.05	**30.27**	42.90	36.80	74.95	73.46	52.14	60.50	77.50	56.91	2.90
0.05	29.17	46.70	38.00	72.68	74.75	50.79	61.50	75.33	56.57	2.75
0.2	28.17	50.50	43.30	76.05	75.92	52.07	61.42	72.94	57.55	2.75
0.4	30.23	54.50	46.40	76.64	76.29	54.07	62.83	74.17	59.14	2.65
0.6	29.83	59.20	49.70	**79.09**	77.42	56.71	58.58	74.33	60.12	3.00
0.8	29.40	60.50	51.70	77.91	**77.63**	58.07	58.25	73.89	60.16	3.05
1	30.03	**67.10**	**53.10**	78.00	77.21	**59.21**	57.42	74.61	**60.95**	**3.50**
					Multi-round Inference					
-1	**62.60**	64.00	54.20	21.59	57.79	62.50	10.83	16.61	44.47	0.75
-0.8	59.63	64.90	61.50	22.27	62.46	61.93	13.17	17.39	45.57	0.75
-0.6	54.65	67.87	65.67	25.74	67.83	59.47	22.75	20.08	47.40	0.65
-0.4	52.77	68.74	64.46	32.30	69.57	61.44	30.78	26.21	49.98	1.05
-0.2	48.73	68.74	62.42	38.67	74.67	59.16	39.00	32.02	52.07	1.25
-0.05	46.48	69.97	67.17	46.38	76.04	60.97	48.79	46.58	56.35	1.40
0.05	45.32	70.08	68.84	51.19	76.62	62.18	52.68	50.47	58.04	1.80
0.2	44.81	69.91	66.73	58.35	80.20	60.34	62.41	63.22	61.79	3.41
0.4	44.30	**72.50**	69.10	64.50	81.75	63.50	68.00	73.56	65.39	5.60
0.6	43.71	68.87	68.20	71.71	83.87	59.83	71.98	81.77	67.47	5.05
0.8	44.35	68.37	68.00	75.94	84.49	61.54	70.31	84.88	68.76	**6.00**
1	44.07	70.90	**69.60**	**81.41**	**85.08**	**64.43**	**72.58**	**87.22**	**70.74**	4.00

Table 1: The overall performance of LLaMA3-8B-Instruct on multi-constraint instructions with different CDDI values. From left to right, we sort the constraint types from the hardest to the easiest.

Round	Length	keywords	language	ChangeCase	Format	Content	Startend	Punctuation	C_level	I_level
1	29.93	73.46	44.40	50.68	76.59	77.11	59.92	34.40	56.01	2.70
2	29.83	73.29	43.80	50.79	73.36	78.17	61.50	32.60	55.49	2.65
3	30.27	73.46	42.90	52.14	74.95	77.50	60.50	36.80	56.91	2.90
	30.01±0.23	73.40±0.10	43.70±0.75	51.20±0.82	74.97±1.61	77.59±0.53	60.64±0.80	34.60±2.11	56.14±0.72	2.75±0.13

Table 2: The performance of LLaMA3-8B-Instruct when given the multi-constraint instruction in different constraint orders while sharing the same CDDI value. By calculation, we obtain the P-value=0.9979.

et al., 2016; Wu et al., 2020). Specifically, we leverage the importance of the input tokens to measure the LLMs' attention to them. To obtain the importance of a specific instruction token t_x to a response token t_y, we calculate the confidence change after the removal of the t_x, as formulated below:

$$I_{t_x,t_y} = p(t_y|Z_y) - p(t_y|Z_{y,/t_x}), \tag{6}$$

where $p(\cdot|\cdot)$ is the conditional probability produced by the LLM f, Z_y is the tokens before the t_y and $Z_{y,/t_x}$ is the tokens of Z_y after removing the token t_x. To reduce the computation, we approximate the I_{t_x,t_y} with the first-order gradient $\frac{\partial f(t_y|Z_y)}{\partial \mathbf{E}[t_x]}$ (Wu et al., 2023), where $\mathbf{E}[t_x]$ is the token embedding of t_x. We normalize the importance I_{t_x,t_y} and obtain the standard importance S_{t_x,t_y} with the formula:

$$S_{t_x,t_y} = \frac{L \times I_{t_x,t_y}}{\max_{i=1}^{N_X} I_{t_i,t_y}}, \tag{7}$$

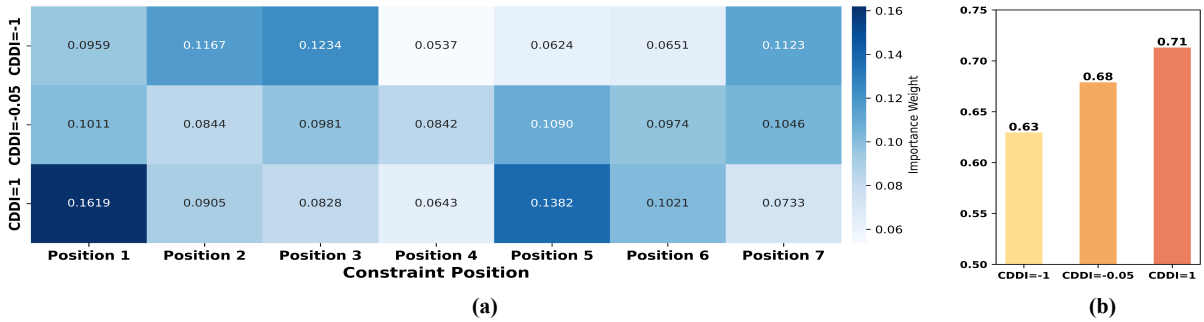
where N_X is the number of instruction tokens and L is a hyper-parameter which helps to filter the noise brought by the first-order approximation. To visualize the LLMs' attention to different constraints, we calculate the importance weight of a specific constraint C_x to the final response Y with the formula:

$$S_{C_x,Y} = \frac{1}{N_Y} \sum_{t_y \in Y} \sum_{t_x \in C_x} S_{t_x,t_y}, \tag{8}$$

where N_Y is the number of response tokens.

5.2 Experiment Set-up

We conduct our explanation study on the LLaMA3-8B-Instruct model. We set the hyper-parameter L to 10 in Eq.(7) and select three most typical difficulty distributions: hard-to-easy (indicated by CDDI=1), easy-to-hard (indicated by CDDI=-1) and random (indicated by CDDI=-0.05) to conduct our experiments. We randomly sample 200



(a) (b)

Figure 6: (a) The importance weights assigned by the LLM when handling constraints in different positions. (b) The total importance weights which designated to the constraint part in the multi-constraint instructions among three different constraint distributions.

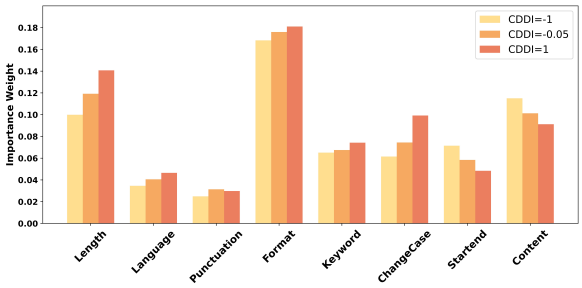


Figure 7: The importance weights across different types of constraint in three different constraint distributions.

instances from the corresponding data which fall in the required CDDI value in the probing task to serve as the dataset.

5.3 Results

Hard-to-easy constraint order induces the LLM to pay more attention to the constraint part in the multi-constraint instructions. We visualize the importance weights of the model on the constraints in different positions. As shown in Fig. 6 (a), in the multi-constraint instruction following, the model's attention on different positions varies with changes in the constraint orders. Specifically, when the constraints are randomly distributed across different positions (represented by CDDI=-0.05), the model assigns similar attention to all positions. As the constraint order becomes more structured (represented by CDDI=-1 and CDDI=1), the model's attention neither exhibits the "lost in the middle" phenomenon observed in long-context processing (Liu et al., 2024), nor a simply sequential distribution, but follows an iterative, laddered order. Then, in Fig. 6 (b), we present the total importance weight the model assigns to the constraint part. We observe that the "hard-to-easy" constraint order attracts the most attention from the model towards the constraint part, which provides an explanation for the superiority of this constraint order.

The LLM's performance on various constraints is strongly correlated with its attention patterns. The importance weights of the model on different types of constraints are presented in Fig. 7. Among the three distinct difficulty distributions, the "hard-to-easy" (represented by CDDI = 1) assigns the highest importance weights to various types of constraints except for the Content and Startend. It is worth noting that this is exactly in accord with quantitative results in Tab. 1, i.e., as the CDDI value increases, the model's performance on the Content and Startend constraints shows a decreasing trend instead. Overall, the results show that the model's accuracy in following a specific type of constraint is strongly correlated with the attention assigned to it by the model.

6 Conclusion

In this paper, we systematically investigate the position bias problem in the multi-constraint instruction following. To quantitatively measure the disparity of constraint order, we propose a novel Difficulty Distribution Index (CDDI). Based on the CDDI, we design a probing task. First, we construct a large number of instructions consisting of different constraint orders. Then, we conduct experiments in two distinct scenarios. Extensive results reveal a clear preference of LLMs for "hard-to-easy" constraint orders. To further explore this, we conduct an explanation study. We visualize the importance of different constraints located in different positions and demonstrate the strong correlation between the model's attention distribution and its performance.

7 Limitations

Our work mainly focuses on the position bias problem in the multi-constraint instruction following. We make a quantitative analysis of the influence brought by different constraint orders in the instructions. However, there are still some limitations. The constraints in our work are usually parallel to each other, which means the order change will not affect the semantic meaning of the instructions. The position bias problem for for those sequential constraints need to be further explored. Moreover, we only investigate the phenomenon of position bias in existing LLM without offering a solution. In further work, we will conduct a further probing task in sequential constraints to improve the generalization of our findings.

References

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

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Lukas Berglund, Meg Tong, Max Kaufmann, Mikita Balesni, Asa Cooper Stickland, Tomasz Korbak, and Owain Evans. 2023. The reversal curse: Llms trained on" a is b" fail to learn" b is a". *arXiv preprint arXiv:2309.12288*.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, C. Monz, Arianna Bisazza, and M. D. Rijke. 2024. The SIFo Benchmark: Investigating the Sequential Instruction Following Ability of Large Language Models. In *Conference on Empirical Methods in Natural Language Processing*, volume abs/2406.19999, pages 1691–1706.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. Premise order matters in reasoning with large language models. In *Forty-first International Conference on Machine Learning*.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-Play with Tree-Search Refinement to Improve Instruction-Following in Large Language Models. *arXiv*, abs/2412.11605.

Vincent A Cicirello. 2020. Kendall tau sequence distance: Extending kendall tau from ranks to sequences. *EAI Endorsed Transactions on Industrial Networks and Intelligent Systems*, 7(23).

Vittoria Dentella, Fritz Günther, Elliot Murphy, Gary Marcus, and Evelina Leivada. 2024. Testing ai on language comprehension tasks reveals insensitivity to underlying meaning. *Scientific Reports*, 14(1):28083.

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*.

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

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

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

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richárd Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in Neural Information Processing Systems*, 36.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. 2016. Visualizing and understanding neural models in nlp. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 681–691.

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

Junjie Liu, Shaotian Yan, Chen Shen, Liang Xie, Wenxiao Wang, and Jieping Ye. 2023. Concise and organized perception facilitates reasoning in large language models. *arXiv e-prints*, pages arXiv–2310.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

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

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

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adri Garriga-Alonso, et al. 2023. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*.

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

Raphael Tang, Crystina Zhang, Xueguang Ma, Jimmy Lin, and Ferhan Türe. 2024. Found in the middle: Permutation self-consistency improves listwise ranking 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 2327–2340.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Mengting Wan, Tara Safavi, Sujay Kumar Jauhar, Yujin Kim, Scott Counts, Jennifer Neville, Siddharth Suri, Chirag Shah, Ryen W White, Longqi Yang, et al. 2024. Tnt-llm: Text mining at scale with large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 5836–5847.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *The 61st Annual Meeting Of The Association For Computational Linguistics*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *Preprint*, arXiv:2204.07705.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxing Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Benchmarking Complex Instruction-Following with Multiple Constraints Composition. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, volume abs/2407.03978.

Xuansheng Wu, Wenlin Yao, Jianshu Chen, Xiaoman Pan, Xiaoyang Wang, Ninghao Liu, and Dong Yu. 2023. From language modeling to instruction following: Understanding the behavior shift in llms after instruction tuning. *arXiv preprint arXiv:2310.00492*.

Z Wu, Y Chen, CM Kao, and FUN Liu. 2020. Perturbed masking: Parameter-free probing for analyzing and interpreting bert. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*. Association for Computational Linguistics.

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

Tianyi Yan, Fei Wang, James Y. Huang, Wenxuan Zhou, Fan Yin, A. Galstyan, Wenpeng Yin, and Muhao Chen. 2024. Contrastive Instruction Tuning. In *Annual Meeting of the Association for Computational Linguistics*, volume abs/2402.11138.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Wenpeng Yin, Qinyuan Ye, Pengfei Liu, Xiang Ren, and Hinrich Schütze. 2023. Llm-driven instruction following: Progresses and concerns. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: Tutorial Abstracts*, pages 19–25.

Weizhe Yuan, Ilia Kulikov, Ping Yu, Kyunghyun Cho, Sainbayar Sukhbaatar, Jason Weston, and Jing Xu. 2024. Following length constraints in instructions. *arXiv preprint arXiv:2406.17744*.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. 2023. Evaluating Large Language Models at Evaluating Instruction Following. In *NeurIPS Workshop on Instruction Tuning and Instruction Following*, volume abs/2310.07641.

Xianren Zhang, Xianfeng Tang, Hui Liu, Zongyu Wu, Qi He, Dongwon Lee, and Suhang Wang. 2024a. Divide-Verify-Refine: Aligning LLM Responses with Complex Instructions. *arXiv*, abs/2410.12207.

Zhen Zhang, Yuhua Zhao, Hang Gao, and Mengting Hu. 2024b. Linkner: Linking local named entity recognition models to large language models using uncertainty. In *Proceedings of the ACM on Web Conference 2024*, pages 4047–4058.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 46595–46623.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023a. Instruction-Following Evaluation for Large Language Models. *arXiv*, abs/2311.07911.

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

A Appendix

A.1 Implementation Details

We utilize 8 NVIDIA A800 80GB GPUs to conduct all the experiments. We employ the vLLM framework (Kwon et al., 2023) to accelerate the model inference. For reproducibility, we employ the greed search in the whole inference (i.e., setting the "do_sample" to false.).

A.2 More details for Comstraint Sampling

In this work, We categorize the constraints into 8 different groups. The categorization is shown in the Tab. 3. For each group, there are multiple types of constraints. Specifically, the constraints are designated to: (1) Keyword constraints. These constraints focus on controlling the inclusion or exclusion of specific words or phrases within the response. (2) Language constraints. Language constraints govern the linguistic properties of the response, including the language in which the response is written (e.g., English). (3) Length constraints. These constraints focus on controlling the overall length of the response, including the number of paragraphs, words, and sentences. (4) Content Constraints. Content-related constraints define additional rules to ensure the response contains specific elements. (5) Format constraints. Formatting constraints focus on how the response is structured and styled. For example. (6) ChangeCase Constraints. These constraints focus on adjusting the case of words in the response. They may require the entire response to be in uppercase letters (e.g., ALL CAPS), or entirely in lowercase letters (e.g., all lowercase). (7) StartEnd constraints. These constraints limit the very beginning or ending of the model outputs. (8) Punctuation constraints. These constraints limit the appearance of specific commas.

Considering the LLM is vulnerable to different descriptions of the constraints (Yan et al., 2024), we employ the GPT4o-mini to generate different descriptions of the same constraints. Specifically, given a description example, we leverage the prompt shown in the Tab. 4 to seven distinct variants. Overall, we obtain 8 distinct descriptions for a specific type of constraint.

Constraint Group	Constraint	Description Example
Keyword	Include Keywords	Include keywords [keyword1], [keyword2] in your response.
	Exclude Keywords	Do not include keywords [forbidden words] in the response.
	Keyword Frequency	In your response, the word should appear N times.
	Letter Frequency	In your response, the letter [letter] should appear [N] times.
Language	Response Language	Your ENTIRE response should be in [language], no other language is allowed.
Length	Number Paragraphs	Your response should contain [N] paragraphs. You separate paragraphs using the markdown divider ***.
	Number Words	Answer with at least/around/at most [N] words.
	Number Sentences	Answer with at least/around/at most [N] sentences.
	Number Paragraphs + First Word in i-th Paragraph	There should be [N] paragraphs. Paragraphs and only paragraphs are separated with each other by two line breaks. The [i]-th paragraph must start with [first_word].
Content	Postscript	At the end of your response, please add a postscript starting with [postscript marker].
	Number Placeholder	The response must contain at least [N] placeholders representing the word space brackets, such as [address].
Format	Number Bullets	Your response must contain exactly [N] bullet points. Use the markdown bullet points such as: * This is a pont.
	Title	Your answer must contain a title, wrapped in double angular brackets, such as «option of joy».
	Choose From	Your response should contain one of the following options: [options].
	Minimum Number Highlighted Section	Highlight at least [N] sections in your answer with markdown, i.e. *highlighted section*.
	Multiple Sections	Your response must have [N] sections. Mark the beginning of each section with [section_splitter] X.
	JSON Format	Entire output should be wrapped in JSON format.
ChangeCase	All Uppercase	Your entire response should be in English, capital letters only.
	All Lowercase	Your response should be in English, and in all lowercase letters. No capital letters are allowed.
	Frequency of All-capital Words	In your response, words with all capital letters should appear at least [N] times.
StartEnd	End Checker	Your response must finish with this phrase: <end_phrase>.
	Quotation	Wrap uour entire response with double marks.
Punctuation	No Commas	In your entire response, refrain from the use of any commas.

Table 3: The categorization for different constraints.

/ Task prompt */*
You are provided with a <constraint> in an instruction. As a prompt engineer, your task is to rephrase the provided <constraint> to make it more diverse. You ought to provide five more variants of the <constraint>. Make sure your revision does not change the meaning of the original <constraint>.

/ Example */*
—INPUT—
<constraint>:
Your response should contain at least 3 sentences.
—OUTPUT—
variants:
1. Respond with at least three sentences
2. Use at least 3 sentences in your reply
3. Your entire response should include at least three sentences
4. Organize your entire response in at least 3 sentences
5. Please make sure the response is at least 3 sentences long

/ Input */*
—INPUT—
<constraint>:
{**Given_constraint**}
—OUTPUT—
variants:

Table 4: The prompts for diversifying the descriptions of a given constraint. We utilize one-shot in-context learning to enhance the performance. The information that requires manual input is highlighted.