

Constraint Back-translation Improves Complex Instruction Following of Large Language Models

Yunjia Qi*

qyj23@mails.tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Hao Peng*

peng-h24@mails.tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Xiaozhi Wang

wangxz20@mails.tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Bin Xu

xubin@tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Lei Hou

houlei@tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Juanzi Li[†]

lijuanzi@tsinghua.edu.cn
Department of Computer Science and
Technology, Beijing National
Research Center for Information
Science and Technology
Tsinghua University
Beijing, China

Abstract

Large language models (LLMs) struggle to follow instructions with complex constraints in format, length, etc. Following the conventional instruction-tuning practice, previous works conduct post-training on complex instruction-response pairs generated by feeding complex instructions to advanced LLMs. However, even advanced LLMs cannot follow complex instructions well, thus limiting the quality of generated data. In this work, we find that *existing datasets inherently contain implicit complex constraints* and propose a novel data generation technique, *constraint back-translation*. Specifically, we take the high-quality instruction-response pairs in existing datasets and only adopt advanced LLMs to add complex constraints already met by the responses to the instructions, which naturally reduces costs and data noise. In the experiments, we adopt Llama3-70B-Instruct to back-translate constraints and create a high-quality complex instruction-response dataset, named CRAB. We present that post-training on CRAB improves multiple backbone LLMs' complex instruction-following ability, evaluated on extensive instruction-following benchmarks. We further find that constraint back-translation also serves as a useful auxiliary training objective in post-training. Our code, data, and models are released to facilitate future research¹.

*Equal contribution.

[†]Corresponding Author.

¹<https://github.com/THU-KEG/Crab>

CCS Concepts

• Computing methodologies → Natural language processing.

Keywords

Post-Training, Instruction-Following, Large Language Models

ACM Reference Format:

Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2025. Constraint Back-translation Improves Complex Instruction Following of Large Language Models. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25)*, November 10–14, 2025, Seoul, Republic of Korea. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3746252.3761324>

1 Introduction

Large language models (LLMs) have achieved remarkable performance in numerous natural language processing tasks [13, 37, 50, 57, 61]. However, they still fall short in following instructions with complex constraints [24, 44, 62], such as length constraints shown in Figure 1, which limits their effectiveness and usability.

To enhance the instruction-following ability of LLMs, the standard practice is to post-train the targeted LLM on a large set of instruction-response data pairs. For the complex instruction-following with multiple constraints, existing efforts [18, 48] synthesize complex datasets by adding multiple constraints to existing instructions and generating responses with advanced LLMs like GPT-4 [37]. While this data generation pipeline is straightforward and widely adopted, even the most capable LLMs cannot follow complex instructions well [24, 44], which limits the quality of generated data and necessitates laborious filtering. The status quo urges the development of automatic data generation methods relying less on existing LLMs' complex instruction-following abilities.

Our key observation is that *existing datasets inherently include implicit complex constraints* so that we can reuse the widely-available



This work is licensed under a Creative Commons Attribution 4.0 International License. *CIKM '25, Seoul, Republic of Korea*

© 2025 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-2040-6/2025/11

<https://doi.org/10.1145/3746252.3761324>

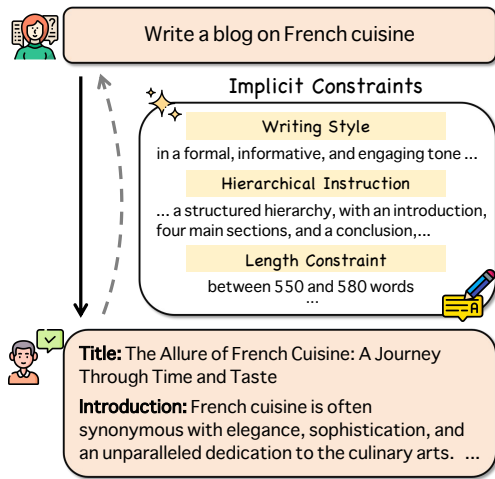


Figure 1: Existing datasets inherently include implicit satisfied complex constraints in the responses.

high-quality instruction-following datasets [26, 35, 49, 56] to synthesize complex instruction-response pairs. As shown in Figure 1, although the original concise instruction does not explicitly specify constraints like writing style or length, the response already satisfies some constraints in multiple dimensions. Therefore, we can efficiently create high-quality complex instruction-response pairs from existing datasets by generating constraints from responses and adding them to instructions. We dub this data generation method as *constraint back-translation*. It only requires discovering the constraints already met by responses rather than following the complex instructions with multiple constraints, which significantly reduces the requirements for model capability. As a result, it is both cost-effective and capable of producing high-quality data with limited noise. We also find that constraint back-translation can serve as a useful auxiliary training objective in post-training, dubbed as the *reverse training* technique. Specifically, we use instructions and responses as inputs to train the model to output constraints in post-training. The intuition is that reverse training may enhance the model’s understanding of constraints and improve its efficacy [17].

We adopt Llama3-70B-Instruct [13] to back-translate constraints from a collection of existing data, generating a large-scale complex instruction-following dataset, named CRAB. Specifically, we sample a total of 13,500 instances from existing high-quality instruction-following datasets [15, 26, 39, 56] as the seed data, and manually define a scope of common constraints. We then use the original instruction, response, and constraint scope as inputs to Llama3-70B-Instruct to generate the corresponding implicitly satisfied constraints. Following previous works [18, 48], we train the LLMs using the mixture of CRAB and ShareGPT dataset [9], and we jointly adopt standard supervised fine-tuning and reverse-training on CRAB. In the experiments, we select the capable open-source LLMs Llama3 8B [13] and Mistral 7B [23] as backbone models and evaluate the complex instruction-following abilities of our models against various baselines on IFEval [62] and FollowBench [24]. The results demonstrate that training on CRAB significantly enhances LLM

performance in complex instruction following. We also conduct evaluation for general instruction-following abilities on AlpacaEval [29] and find that our models achieve even greater improvements over prior work on complex instruction-following, such as Conifer [48]. This indicates that constraint back-translation yields higher general data quality than previous techniques relying on the ability of advanced LLMs. Ablation studies further validate the efficacy of our CRAB dataset and reverse training approach. Finally, we discuss the advantages and challenging scenarios for our constraint back-translation method with experiments.

In summary, our contributions are threefold: (1) We propose constraint back-translation, a cost-effective and high-quality data construction method for complex instruction following. (2) We construct CRAB, a high-quality complex instruction-following dataset, and design a reverse training method for developing Llama3_{CRAB}, Mistral_{CRAB} models with better complex instruction-following abilities. (3) We conduct extensive experiments to demonstrate the efficacy of CRAB and discuss key design choices and potential improvement opportunities to inspire future research on following complex instructions with multiple constraints.

2 Method

This section introduces the construction process of CRAB (§ 2.1) and the training method (§ 2.2).

2.1 Constructing CRAB

We begin by introducing the notions. Given an instruction x , which typically defines a specific task, such as “*Write a blog on French cuisine*”, a set of constraints c , which specify conditions for the response, such as length restrictions, and a response y that satisfies both the constraints c and the instruction x , our goal is to construct a high-quality dataset of (x, c, y) triples. We first collect a set of high-quality (x, y) pairs from existing datasets and then apply constraint back-translation to generate the constraints c for each (x, y) pair. The data construction process is illustrated in Figure 2, which consists of three steps: data collection, constraint back-translation, and constraint combination. In the data collection process, we collect a comprehensive set of high-quality (x, y) pairs from existing datasets. We then back-translate the corresponding c for each (x, y) using Llama3-70B-Instruct and Python scripts automatically. Finally, we perform filtering and a combination of the constraints c to construct CRAB.

Data Collection. We first collect a comprehensive set of (x, y) as seed data from four existing widely-used high-quality supervised fine-tuning datasets, including Alpaca GPT4 [39], Orca Chat [15], Evol Instruct [56], and OpenAssistant [26]. These datasets are of high quality and contain diverse data instances for generating rich constraints. For Alpaca GPT4 and OpenAssistant, which contain the human annotated quality score for each instance, we use the instances with the highest quality. Moreover, to ensure that the responses include diverse constraints implicitly, we only consider instances where the response exceeds 200 words. From each dataset, we randomly sample the qualified instances using an examples-proportional mixture strategy [53], with a maximum of 4,500 instances per dataset.

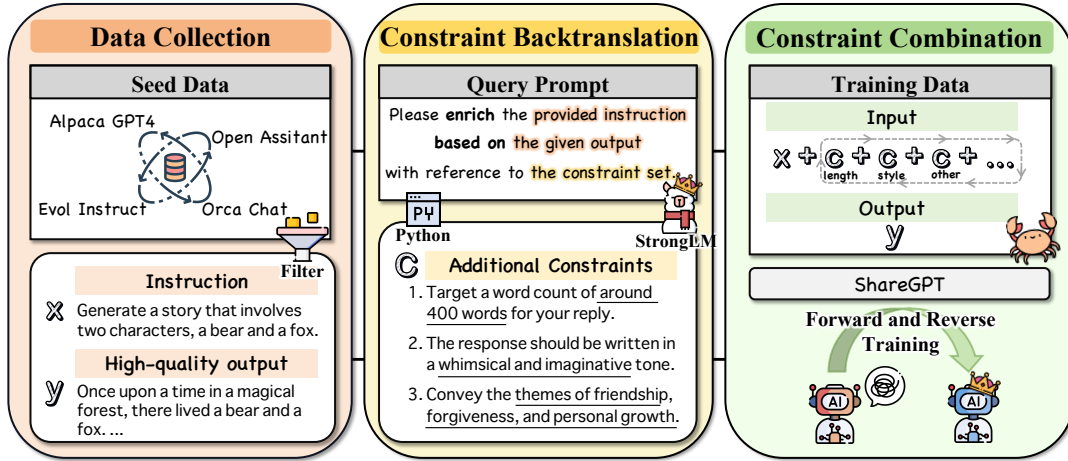


Figure 2: The framework of constructing the proposed alignment training dataset.

Constraint Back-translation. We adopt the Llama3-70B-Instruct LLM and Python scripts to back-translate constraints for the seed data. Specifically, we use Llama3-70B-Instruct to automatically generate constraints implicitly satisfied by the response from instruction-response pairs. To guide the generation process and encourage diversity, we manually select 13 commonly used constraint types² as in-context examples in the prompt. These examples help the model not only reproduce the same constraint types but also generalize to uncover new constraint types beyond the original 13. In total, this process results in over 100 distinct constraint types, with the 13 example types appearing more frequently. We then use Llama3-70B-Instruct to re-verify whether the response satisfies the generated constraints and exclude the constraints that are not met. Considering some constraints, e.g., length constraint, cannot be effectively followed by LLMs [48], leading to noisy back-translation, and some constraints can be easily generated using Python scripts, we choose to adopt Python scripts for 6 types of constraints. Specifically, we write and paraphrase several templates for each of these constraints. For example, for a length constraint, one template is “Please generate a response with fewer than <placeholder> words but more than <placeholder> words”. We then use Python scripts to automatically identify the value for this constraint in the response and fill the templates to construct a constraint. For number-related constraints such as length, we use NLTK [32] to compute relevant statistics from the response (e.g., word count), and then randomly sample a range that includes the computed value to complete the template. For keyword constraints, we apply YAKE [5], a lightweight unsupervised keyword extraction method, to extract the top 3 most significant keywords from the response to form relevant constraint content. For punctuation constraints, we randomly select punctuation marks based on the response and use them to populate the corresponding templates. We adopt ROUGE-L [30] with a threshold of 0.6 to exclude similar constraints. Finally, we sample and manually review 50 instruction-response pairs and

their generated constraints, finding minimal noise and high compliance between constraints and response. Table 1 reports the 19 most representative constraint types, comprising the 13 prompt-based categories and 6 Python-generated ones, along with their frequency of occurrence in CRAB. Moreover, for 13 types of constraints generated by the LLM, we analyzed the keywords of each generated constraint and extracted the top 10 keywords for each major category, presenting them as subcategories in Table 2.

Constraint Combination. Finally, we combine individual constraints to form the final constraint c for each instruction. Previous studies have shown that increasing the number of constraints in the training data leads to better model performance [18]. Therefore, we enhance each instruction with a combination of multiple constraints. Specifically, we randomly sample 6 to 8 constraints from each instruction’s constraint set generated in the previous step, shuffle their order, and recombine them into the final constraint c . Similar to previous work [43, 48], we add 1 to 3 in-context demonstrations for 50% of the data. Finally, we construct CRAB with 13,500 instances, containing an average of 7.1 constraints.

2.2 Model Training

To further enhance LLMs’ understanding of complex constraints, we propose a *reverse training* method that takes the instruction-response pair (x, y) as input to teach LLMs to generate the constraints c . The intuition is that correctly generating constraints requires sufficient comprehension first. Formally, the reverse training objective is to minimize $\mathcal{L}_r$, where $\mathcal{L}_r = -\log P_\theta(c|x, y)$ and the LLM is parameterized by θ . We also adopt the standard supervised fine-tuning (SFT; 38), named *forward training*, to minimize $\mathcal{L}_f$, where $\mathcal{L}_f = -\log P_\theta(y|x, c)$. The final training objective is a combination of $\mathcal{L}_f$ and $\mathcal{L}_r$: $\mathcal{L} = \alpha\mathcal{L}_f + (1 - \alpha)\mathcal{L}_r$. We set α to 0 during 70% of the training process, and to 1 for the remaining time. We train the LLM using a mixture of ShareGPT [9] and CRAB. We adopt $\mathcal{L}_f$ when training on ShareGPT [9] and adopt $\mathcal{L}$ on CRAB. We train a base LLM on this data and obtain the SFT version of our model. Based on the SFT trained model, we continue training using

²We sample 200 instruction-response pairs from ShareGPT to observe real-world constraint needs and summarize them.

Table 1: Constraint types defined during the back-translation process. The Generator indicates the component responsible for extracting the implicit constraints from the response. “LLM” denotes model-inferred constraints; “Python” denotes rule-extracted ones. The Rate (%) indicates the proportion of data instances in which constraints of this type are extracted.

Constraint Category	Description	Generator	Rate
Situation	Adding conditions, clarifying the subject or object, or defining the circumstances under which the instruction applies.	LLM	36.9
Writing Style	Specify the style requirements for the response to align with the intended message and audience.	LLM	81.3
Semantic Elements	Clearly articulate the main theme, focus, meaning, or underlying concept of the response.	LLM	99.5
Morphological	Outline specific prohibitions, such as avoiding certain words or phrases and refraining from specific formatting styles.	LLM	99.7
Multi-lingual	Specify the language(s).	LLM	94.8
Literary Devices	Identify any particular literary devices to be employed.	LLM	91.7
Grammatical Structure	Specify the grammatical structure.	LLM	99.1
Hierarchical Instructions	Establish a response hierarchy, defining the prioritization and structuring of tasks within the output.	LLM	83.1
Output Format	Depending on the required format of the output—such as Python, tables, JSON, HTML, LaTeX—impose relevant format constraints.	LLM	15.2
Paragraphs Constraints	Clearly specify the required number of paragraphs or sections in the text. Additionally, indicate any specific spacing or separators needed—such as blank lines, horizontal rules, or special symbols to enhance readability and visual appeal.	LLM	71.3
Specific Sentence	Specify a particular phrase to be included either at the beginning or end of the text, clearly indicating its exact placement.	LLM	70.1
Header Format	Specify the formatting style for titles or keywords within the Output, such as using bold, italics, or CAPITAL LETTERS.	LLM	9.5
Item Listing Details	Clearly specify the formatting for individual entries within the text. Direct the use of specific symbols for listing—such as bullet points (•), numbers (1., 2., 3., etc.), or hyphens (-).	LLM	67.7
Length Constraint	Determine the word count of the output text to establish length constraints.	Python	47.6
Word Constraint	Determine the number of words in each sentence to set word constraints.	Python	15.1
Sentence Constraint	Determine the number of sentences in each paragraph to establish sentence constraints.	Python	20.6
Character Constraint	Determine the number of characters in each word.	Python	21.4
Keyword Constraint	Determine the keywords in the output text to make the constraints more detailed.	Python	45.1
Punctuation Limitation	Specify which punctuation marks cannot be used in the output text.	Python	21.6

the Direct Preference Optimization objective (DPO; 45). Specifically, same as the DPO phrase by Sun et al. [48], we use the high-quality DPO dataset UltraFeedback [12] to conduct further training and obtain the DPO version of our model.

We conduct all experiments using 8 NVIDIA A100 80GB GPUs and apply DeepSpeed ZeRO Stage 2 for both SFT and DPO training. The implementation of the ratio α is achieved by segregating the dataset into two parts, since the model may memorize the data during the forward process. In the SFT stage, we set the learning rate to 5×10^{-6} , with a per-device batch size of 4 and 8 gradient accumulation steps. The warm-up ratio is set to 0.1. The Mistral 7B experiments are trained for 4 epochs with a maximum sequence length of 2048, while the Llama 3 8B experiments are trained for 3 epochs with a maximum sequence length of 4096. For DPO training, we set the learning rate to 5×10^{-7} , with a per-device batch size of 4 and 2 gradient accumulation steps. The maximum sequence

length is fixed at 2048. For Mistral 7B, we train for 1 epoch with a cosine learning rate schedule and set the DPO beta to 0.01. For Llama 3 8B, we train for 3 epochs with a linear scheduler and set the beta to 0.1, following Ivison et al. [22].

3 Experiments

In this section, we introduce the experimental setup (§ 3.1), experimental results (§ 3.2), and further analyses (§§ 3.3 to 3.6).

3.1 Experimental Setup

Backbone Models. We adopt two widely-used open-source base models, Mistral 7B [23] and Llama 3 8B [13], as our backbone models for developing Llama3_{C_{RAB}} and Mistral_{C_{RAB}}. Specifically, we employ Mistral-7B-v0.3 and Meta-Llama-3-8B, downloaded from Hugging Face [55]. During the SFT stage, we adopt a 5×10^{-6} learning rate, 256 batch size, and train the Mistral for 4 epochs and

Table 2: Top-10 subcategories for each major constraint type (excluding *Situation*) identified via constraint back-translation, as the *Situation* type is closely tied to the input and lacks clustering tendencies.

Constraint Type	Top-10 Subcategories
Morphological	technical jargon, complex vocabulary, using contractions, mathematical formulas, colloquial, promotional language, ambiguous, first-person narrative, medical terminology, financial terminology
Writing Style	informative, objective, concise, technical, empathetic, analytical, professional, instructional, imaginative, encouraging
Semantic Elements	main theme, key messages emphasizing, highlighting the importance, providing detailed information, natural language processing, providing actionable tips, providing accurate information, artificial intelligence, climate change, product review
Multi-lingual	English, Spanish, Russian, French, German, Chinese, Catalan, Italian, Turkish, Finnish
Grammatical Structure	logical structure, convey nuanced ideas, maintain reader engagement, transitional phrases, imperative mood, person singular, coherent structure, past tense, consistent verb tense, chronological structure
Hierarchical	JSON, signaling cascades, data analysis, main sections, function logic, Python, logical structure, prioritize the explanation, narrative, root element
Output Format	plain text format, Python code format, written article format, clear subject line, formal business email, Markdown format, numbered list format, HTML, triple backticks, programming language format
Paragraphs	blank line separating, separators required, clear topic sentence, concluding summarizing, introducing, single code block, providing additional information, special separators, clear heading, HTML code
Specific Sentence	response should begin, include the sentence, answers the question, follows these steps, rephrase the question, final paragraph, sets the scene, detailed explanation, acknowledging the request, concluding sentence summarizing
Item Listing Details	numbered lists, blank line separating, individual entries, listed using hyphens, bullet points, specific symbols required, Arabic numerals, separated by commas, lowercase letters, uppercase letters
Header Format	titles and keywords, double quotation marks, capital letters, bold font, code snippets, plain text, separated by commas, triple backticks, quotation marks, major word capitalized
Literary Devices	technical concepts, transitional phrases, encourage critical thinking, rhetorical questions, descriptive language, empty metaphors, illustrate key points, active voice, rhetorical devices, vivid imagery

Llama 3 for 3 epochs. During the DPO optimization stage, we adopt 5×10^{-7} learning rate, 64 batch size, and 1 training epoch with beta 0.01 for Mistral and 3 epochs with beta 0.1 for Llama 3.

Baselines. Our baselines include popular open-source and proprietary LLMs, divided into three main categories for comparison: (1) Proprietary LLMs, including GPT-3.5 [36] and GPT-4 [37]. (2) General instruction-tuning LLMs, including Vicuna-V1.5 13B [9], trained on the 125k ShareGPT dataset, WizardLM-V1.2 13B [56], trained on the 196k Evol-Instruct dataset, Zephyr beta 7B [52], trained with the UltraFeedback [12] dataset using the DPO objective [45], and Mistral-Instruct 7B v0.3 [23], which achieves leading performance on chat benchmarks based on Mistral 7B (3) Models specifically optimized for complex instruction-following tasks, including Suri-I-ORPO [41], which is optimized for multi-constraint instruction-following tasks in long-form text generation, and the Conifer series [48], which are trained on the data where the constraints, instructions, and responses are all generated using GPT-4.

Evaluation Datasets. We use two widely-used and challenging complex instruction-following datasets IFEval [62] and FollowBench [24] for evaluation. IFEval consists of 541 instructions that can be automatically validated using Python scripts. Each instruction contains 1 to 3 constraints, primarily focusing on strict lexical and formatting constraints. FollowBench is a fine-grained, multi-constraint instruction-following benchmark and it categorizes the

difficulty into five levels (L1 to L5) based on the number of constraints of an instruction, where L1 represents the simplest level with only one constraint, while L5 is the most difficult, with a combination of five constraints. It also includes five constraint categories, including *content*, *situation*, *style*, *format*, and *example*, along with a *mixed* constraint category that combines various categories of constraints. FollowBench contains a total of 820 instructions across more than 50 different NLP tasks, and it is automatically evaluated using either Python scripts or GPT-4. Please refer to the original paper for more details [24].

3.2 Experimental Results

The experimental results are presented in Table 3. Our observations are as follows: (1) After training on the CRAB dataset, our models significantly outperform the corresponding base models and the open-source models trained through SFT on general instruction-following datasets. Our DPO version of models achieves the best performance among the compared models. It demonstrates the effectiveness of our data and training approach. (2) Our models surpass Conifer Sun et al. [48], which is specifically trained for complex instruction-following, on IFEval. It suggests that our model performs better in following lexical and format constraints. However, our models slightly lag behind Conifer on FollowBench. We provide an in-depth discussion on the performance across different constraint categories in FollowBench in § 3.5. We observe that the performance decline is primarily due to the style constraint, where

Table 3: Experimental results (%) of the LLMs on IFEval and FollowBench. In IFEval, “[S]” and “[L]” denote strict and loose accuracy, “P” and “I” indicate the prompt and instruction level. In FollowBench, L1 (simplest) to L5 (hardest) denote different difficulty levels. We highlight the highest and second-highest scores of open-source LLMs using bold font and underline. † and * means the results are from Sun et al. [48] and He et al. [18].

Model	Backbone	IFEval					FollowBench (HSR)						AVG
		[S]P	[S]I	[L]P	[L]I	AVG	L1	L2	L3	L4	L5	AVG	
GPT-3.5*	GPT	59.0	68.5	64.0	73.6	66.3	80.3	68.0	68.6	61.1	53.2	66.2	66.3
GPT-4†	GPT	76.9	83.6	79.3	85.4	81.3	84.7	76.1	71.3	74.5	62.4	73.8	77.6
Vicuna-v1.5-13B†	Llama2	43.1	53.6	46.6	58.0	50.3	71.2	61.3	48.3	38.0	33.1	50.4	50.4
WizardLM-v1.2-13B	Llama2	43.6	54.4	48.4	59.1	51.4	61.3	51.6	43.3	37.5	29.9	44.7	48.1
Conifer _{SFT} -13B†	Llama2	42.9	53.0	47.5	57.4	50.2	60.5	53.6	<u>48.4</u>	40.7	31.7	47.0	48.6
Zephyr-beta-7B†	Mistral	32.0	46.8	44.9	58.0	45.4	57.6	51.9	41.9	41.4	31.4	44.8	45.1
Mistral _{Instruct} -7B	Mistral	<u>48.6</u>	<u>59.8</u>	<u>53.2</u>	<u>64.3</u>	<u>56.5</u>	57.1	51.5	43.6	41.5	33.2	45.4	50.9
Suri-ORPO-7B	Mistral	47.3	58.0	51.4	62.0	54.7	45.4	41.4	24.2	18.6	15.2	29.0	41.9
Conifer _{SFT} -7B†	Mistral	45.8	57.1	50.8	62.0	53.9	54.3	49.5	49.3	40.8	30.5	44.9	49.4
Conifer _{DPO} -7B†	Mistral	48.1	59.1	52.3	63.3	55.7	60.3	53.6	48.0	47.1	41.0	<u>50.0</u>	<u>52.9</u>
Llama3-8B	Llama3	25.7	36.8	28.1	35.1	31.4	4.8	8.7	8.8	6.0	9.8	7.6	19.5
Llama3 _{CRAB}	Llama3	39.4	50.2	43.8	54.2	46.9	57.5	44.9	34.9	25.2	20.0	36.5	41.7
Llama3 _{CRAB} + DPO	Llama3	40.3	52.0	47.7	58.9	49.7	64.6	49.0	41.6	35.8	<u>36.8</u>	45.5	47.6
Mistral-7B	Mistral	18.5	30.8	19.6	31.9	25.2	14.3	16.6	8.3	5.8	5.5	10.1	17.7
Mistral _{CRAB}	Mistral	47.9	57.3	51.6	61.2	54.5	63.9	<u>54.4</u>	40.1	30.4	27.9	43.3	48.9
Mistral _{CRAB} + DPO	Mistral	49.7	61.5	57.7	68.5	59.3	<u>66.1</u>	53.6	53.4	<u>42.4</u>	31.7	49.4	54.4

Table 4: Winning rate (%) of the investigated LLMs on AlpacaEval 2.0 [29]. “LC” denotes length-controlled [14]. † means the results are sourced from the original leaderboard.

Model	LC WinRate	WinRate
GPT-3.5-turbo-0613†	22.4	14.1
GPT-4-0613†	30.2	15.8
WizardLM-70B†	17.6	14.4
WizardLM-v1.2-13B†	14.5	12.0
Vicuna-v1.5-13B†	10.5	6.7
Zephyr-beta-7B†	13.2	11.0
Conifer _{DPO} -7B†	<u>17.1</u>	<u>11.3</u>
Mistral _{CRAB}	13.3	7.9
Mistral _{CRAB} + DPO	18.1	17.6
(vs.) Conifer _{DPO}	60.6	63.5

our models significantly underperform in this constraint category compared to Conifer. Nonetheless, our models achieve significant improvements in real-world scenarios, i.e., the *mixed* constraint in FollowBench [24], compared to Conifer. (3) Training with the DPO objective consistently improves model performance on both evaluation datasets. In this paper, we focus on constructing high-quality SFT data by constraint back-translation, we leave the development of DPO data for complex instruction-following as future work.

3.3 Analysis on General Instruction Following

The complex instruction-following ability not only involves following complex constraints but also encompasses the basic ability to follow instructions themselves, e.g., “*Write a blog on French cuisine*”, named as general instruction following. In this section, we further evaluate our model’s general instruction-following capability. Given that IFEval and FollowBench primarily focus on evaluating the ability to follow constraints, we adopt another widely-used dataset, AlpacaEval [29], which serves as an easy-to-use and high-quality automatic evaluator for instruction-following ability. Specifically, we use AlpacaEval 2.0, which contains 805 instructions, and use gpt-4-1106-preview as the evaluator to get the final weighted win rate. The evaluation results are presented in Table 4, where the “LC WinRate” represents the length-controlled win rate [14]. The default reference model is gpt-4-1106-preview. We can observe that our model significantly outperforms the baseline model Conifer and even exceeds the performance of the 70B version of WizardLM. We also conduct a head-to-head comparison between our model and Conifer. The LC win rate of our model reaches 60.6, significantly outperforming Conifer, which demonstrates that our model possesses a superior general instruction-following capability.

We conduct a further analysis to explore the potential reasons why our model outperforms Conifer in general instruction-following. The primary difference between our model and Conifer is the data construction process. We utilize *constraint back-translation*, where the response is generated directly from the instruction without constraints. In contrast, Conifer uses instruction and corresponding

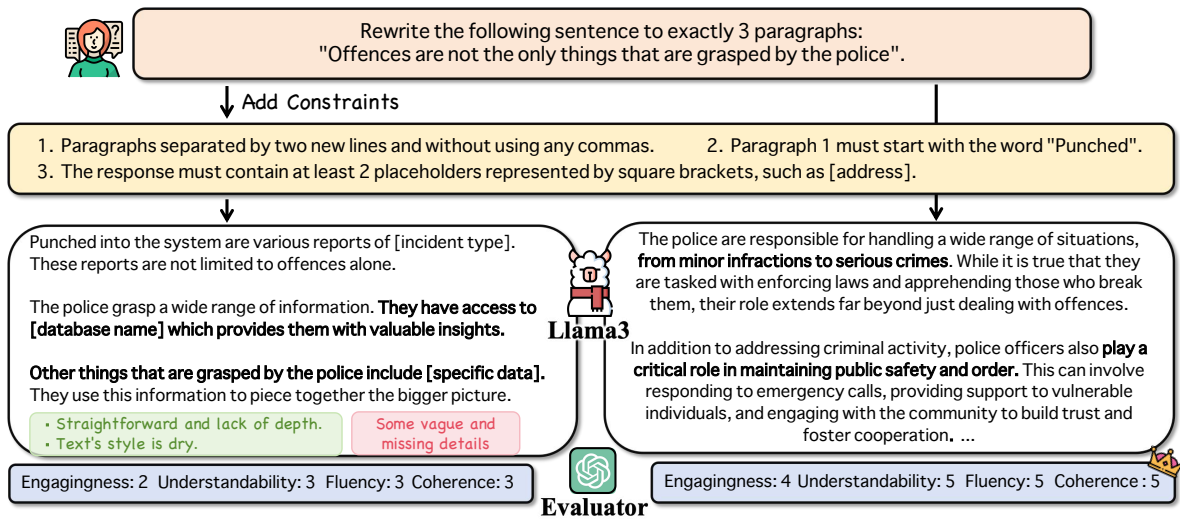


Figure 3: An example of responses generated with and without constraints by Llama3-70B-Instruct. The evaluator is gpt-4o-0806. For better visualization, we present only a subset of the responses generated without constraints.

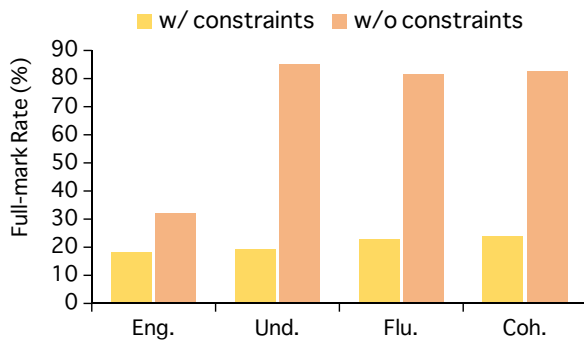


Figure 4: Full-mark rates (%) of the responses generated with and without constraints. The evaluator is gpt-4o-0806, focusing on four widely-used dimensions: Engagingness (Eng.), Understandability (Und.), Fluency (Flu.), and Coherence (Coh.).

constraints to generate the response. We hypothesize that a possible reason is that the response quality in Conifer is lower than CRAB, that is, generating a response conditioned on both instruction and constraints may result in **lower content quality**, such as lower coherence, compared to the response directly generated from instruction without constraints. Intuitively, incorporating constraints may limit the model’s capacity when generating responses. To validate this intuition, we conduct a controlled analytical experiment. Specifically, we sample 100 instruction pairs from FollowBench and IFEval, with each pair consisting of a instruction without constraints and its corresponding multi-constraint version (with over 3 constraints). We first use Llama3-70B-Instruct to generate responses based only on the instructions (w/o constraints). Then, we include the additional constraints and generate corresponding responses (w/ constraints). Following previous work on automated evaluation using advanced LLMs [3, 6], we employ gpt-4o-0806

as the evaluator. We assessing the responses on four dimensions with scores ranging from 1 to 5:

- **Engagingness:** Evaluate how captivating and interesting the text is, based on its ability to hold attention and evoke interest. This includes aspects such as sustaining the reader’s attention, appealing to the intended audience, and creating emotional or intellectual resonance.
- **Understandability:** Evaluate the clarity and ease with which the text can be understood by the target audience. Key factors include the simplicity of expression, appropriateness of language for the audience, and overall clarity of presentation.
- **Fluency:** Evaluate the smoothness of the writing, focusing on grammar, sentence structure, and the natural flow of language. This involves grammatical correctness, varied and coherent sentence construction, and a fluent style.
- **Coherence:** Evaluate the logical flow and consistency of ideas, ensuring the text’s structure is logical and ideas are connected. Considerations include the progression of ideas, effective use of transitions, and internal consistency throughout the response.

We report the full-mark (score 5) rate for each dimension. The results are shown in Figure 4. We can observe that responses generated with constraints significantly underperform those generated without constraints, which suggests that involving constraints when generation may reduce the content quality of the final response. We further conduct a case study, illustrated in Figure 3, showing an example of responses generated with and without constraints. We can find that when involving constraints, the response includes vague terms, such as “[database name]”, and lacks sufficient details and depth. While previous work on complex instruction-following mainly focuses on enhancing the ability to follow multiple constraints, we encourage future work to prioritize response content quality, and constraint back-translation can serve as a solution.

Table 5: Experimental results (%) of the ablation study. *In-Context Demons* denotes in-context demonstrations.

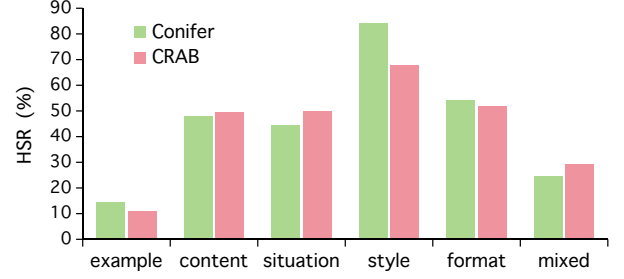
Model	IFEval AVG	FollowBench		AVG
		L1-L2	L3-L5	
Mistral _{CRAB}	54.5	59.1	32.8	48.9
(-) <i>Reverse training</i>	52.1	56.2	33.5	47.3
(-) <i>Forward training</i>	53.9	57.1	32.1	48.0
(-) <i>In-Context Demons</i>	53.6	55.8	30.0	47.0
InstBackT _{SFT}	52.7	55.4	29.3	46.2

Table 6: Experimental results (%) of different reverse training strategies. Keeping forward and reverse data separate (denoted as *separate*, which is our setting) outperforms mixing them during training (denoted as *mixing*).

Model	IFEval AVG	FollowBench		AVG
		L1-L2	L3-L5	
Mistral _{CRAB} (<i>mixing</i>)	53.1	54.9	27.2	45.7
Mistral _{CRAB} (<i>separate</i>)	54.5	59.1	32.8	48.9

3.4 Ablation Study

Main Ablation. We conduct an ablation study to analyze the key factors influencing model performance. Specifically, we investigate three key factors in developing our model: *reverse training*, *forward training*, i.e., standard supervised fine-tuning, and *in-context demonstrations*. We exclude each factor and keep all other conditions identical, to the model separately. When excluding reverse and forward training, we set the loss ratio α in § 2.2 to 1 and 0, respectively. The backbone model is Mistral. The results are presented in Table 5, where L1-L2 in FollowBench represent simpler constraints and L3-L5 denote more complex constraints. We can observe that removing any of these factors leads to a decline in model performance, which demonstrates the effectiveness of these factors in developing our model. For more complex constraints, adding in-context demonstrations during training is effective, as excluding in-context demonstrations leads to a significant performance drop in L3-L5. The reason may be that in-context demonstrations enhance the model’s ability to understand multiple in-context instructions and complex constraints. We further compare with a competitive baseline model, InstBackT_{SFT}, which is trained on synthetic data generated via *instruction back-translation* [28]. The key difference between instruction and constraint back-translation is that the former uses advanced LLMs to generate both instructions and constraints from responses, while the latter focuses on generating constraints from instructions and response pairs. The results are presented in Table 5. We can observe that InstBackT_{SFT} significantly underperforms compared to Mistral_{CRAB}, which suggests that instruction back-translation may produce lower-quality data for complex instruction following. The possible reason is that generating both instructions and constraints simultaneously is more

**Figure 5: Experimental results on different categories of constraints in the FollowBench of Mistral_{CRAB} and Conifer_{SFT}.**

challenging than generating constraints alone. It further demonstrates the efficacy of the constraint back-translation method in creating high-quality training data for improving the complex instruction following ability.

Reverse Training. Intuitively, reverse training enhances the model’s understanding of constraints by requiring it to generate them, as this process inherently demands a deeper comprehension. The implementation of the ratio α between reverse training and forward training is achieved by segregating the dataset into two parts, following Berglund et al. [4], which separates data in their “both” setting. We also experiment with mixing both types of training throughout the process, and the results shown in Table 6 indicate that keeping them separate leads to better model performance. A possible reason is that mixing the forward and reverse training leads to overfitting easily.

3.5 Analysis on Constraint Category

We further investigate our model’s performance across different constraint categories to analyze its strengths and potential limitations. Specifically, we analyze the results on FollowBench, which includes five categories of constraints, including *example*, *content*, *situation*, *style*, and *format*. Please refer to the original paper [24] for the detailed definitions for each constraint category. FollowBench also includes a *mixed* category which is designed for simulating **real-world** scenarios [24], where various types of constraints are combined to form the final constraint. We compare our model Mistral_{CRAB} with the Conifer model, which is trained on the data generated using the standard pipeline: generating the constraints first and then generating the response based on the instruction and constraints. The results on different constraint categories of FollowBench are shown in Figure 5. We can observe that our model significantly outperforms Conifer on the *mixed* constraint, which represents real-world scenarios, suggesting that our model is more effective in handling complex instruction-following scenarios. However, in the *style* constraint category, e.g., “*Write in the style of Shakespeare*”, our model performs significantly worse than Conifer. The possible reason is that the style constraints in our dataset CRAB may be not sufficiently diverse. The data for *style* constraints requires deliberate construction, and the pipeline that generates constraints first and then responses is more effective at generating diverse style constraints, but the responses in our seed data have limited style

Table 7: Experimental results (%) on general benchmarks.

Model	MMLU	GSM8k	MathQA	ARC challenge	ARC easy	Hellaswag	SIQA	AVG
Mistral-7B	62.3	37.2	35.4	48.8	79.5	60.9	46.0	52.9
Mistral _{CRAB}	59.8	46.6	34.4	47.4	77.5	62.5	47.7	53.7
Mistral _{CRAB} + DPO	61.0	34.0	35.7	53.2	79.8	68.1	51.0	54.6

diversity. It suggests a limitation of constraint back-translation, as it relies on diverse responses to generate specific categories of constraints, such as style constraint. Combining the constraint back-translation method with other data generation methods to produce higher-quality data for those specific constraints can further enhance the model’s complex instruction-following ability, and we leave this exploration as future work.

3.6 Analysis on General Abilities

We further assessed our models’ general capabilities using a suite of widely-used benchmarks for: Domain Knowledge Reasoning (MMLU [20]); Mathematical Reasoning (GSM8k [11], MathQA [1]); and Commonsense Reasoning (including HellaSwag [59], SIQA [46], and ARC easy and challenge [10]). As shown in Table 7, improvements in instruction-following capability from constraint back-translation do not degrade general performance. Indeed, performance improved in some cases, such as MathQA, and particularly on commonsense reasoning tasks (ARC challenge, ARC easy, HellaSwag, SIQA), where Mistral_{CRAB} + DPO achieved an average 4.25% improvement. These findings suggest that CRAB can be integrated into existing post-training pipelines to enhance instruction adherence without compromising the model’s general capabilities.

4 Related Work

4.1 Instruction Following

Instruction following task involves following user intentions to generate helpful responses, which is fundamental to modern LLMs [60]. Ouyang et al. [38] first propose the practice of aligning LLMs to follow human instructions, using SFT and RLHF to train models, which is the key factor in the success of ChatGPT [36]. Subsequently, numerous studies focus on enhancing the instruction-following capabilities of LLMs, particularly for open-source models, which can be summarized in two main aspects: (1) data-driven approaches, which design an automated pipeline or use human annotation to produce high-quality training data [2, 9, 28, 31, 35, 39, 40, 43, 49, 56]. (2) new training methods, including novel objectives [16, 19, 34, 45, 63] or training pipelines [8, 28, 52, 58].

A more challenging instruction following scenario is complex constrained instruction following, where the responses should further satisfy specific constraints, such as length, keyword, and format. Previous studies have shown that LLMs struggle to follow these instructions [7, 24, 44]. Recent efforts focus on enhancing this ability by constructing high-quality training data [18, 48]. This process typically involves collecting a set of instructions, constructing constraints, and then generating responses based on the instructions and constraints using advanced LLMs. This work introduces *constraint back-translation*, which generates constraints from

instruction-response pairs, reducing data construction costs and improving data quality by avoiding low-quality responses.

4.2 Back-translation

Back-translation is first proposed in the field of machine translation [21, 47], which mainly is used for data augmentation. It first trains a model to back-translate the target language into the source language, then uses this model to generate parallel training data from a large amount of monolingual target language data, which sufficiently saves human translation efforts. Considering its simplicity and efficacy, back-translation has also been widely applied to tasks like style transfer [42, 51] and paraphrase generation [33, 54]. Its core idea, leveraging model-generated signals to construct training data, makes it a natural fit for self-supervised learning scenarios, including those involving large language models.

Recently, several studies have explored applying back-translation to the field of large language models to efficiently generate high-quality data automatically [25, 27, 41]. Li et al. [27] proposed reversing the training objective to automatically generate corresponding instructions for existing unsupervised corpora, while Pham et al. [41] and Köksal et al. [25] leveraged the powerful general capabilities of LLMs to generate instructions from the corpus directly. Although Pham et al. [41] also generated constraints, it fell within the realm of *instruction back-translation* and did not involve dedicated optimization or exploration for constraint generation. In this work, we propose *constraint back-translation*, an effective data-generation approach that derives high-quality constraints from instruction–response pairs, thereby improving large language models’ ability to follow complex instructions.

5 Conclusion

In this paper, we aim to enhance large language models’ capability for complex constrained instruction following. We propose a *constraint back-translation* data generation method, which can reduce data noise and generation costs, resulting in a high-quality complex instruction-following dataset CRAB. We also propose a *reverse training* method and develop Llama3_{CRAB} and Mistral_{CRAB} based on CRAB. Extensive experiments demonstrate the effectiveness of our data generation method and training method. We further conduct extensive analytical experiments and discuss the key factors, advantages, and potential limitations of our model.

Acknowledgments

This work is supported by National Natural Science Foundation of China (No. 62476150) and, Beijing Natural Science Foundation (No. L243006). The authors also thank all anonymous reviewers and meta-reviewers for their valuable feedback, and the Deng Feng Fund of Tsinghua University for supporting this research.

GenAI Usage Disclosure

In this work, generative AI tools were employed during the data construction phase. As detailed in Section 2.1, we used Llama3-70B-Instruct to generate constraints from instruction-response pairs. Additionally, GPT-4 and Claude were used to refine the phrasing of certain sentences. The authors take full responsibility for the accuracy and originality of all content presented in this paper.

References

- [1] Aida Amini, Saadia Gabriel, Peter Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards Interpretable Math Word Problem Solving with Operation-Based Formalisms. *arXiv:1905.13319* [cs.CL]
- [2] Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models. *arXiv preprint arXiv:2401.18058* (2024).
- [3] Yushi Bai, Jiahao Ying, Yixin Cao, Xin Lv, Yuze He, Xiaozhi Wang, Jifan Yu, Kaisheng Zeng, Yijia Xiao, Haozhe Lyu, et al. 2024. Benchmarking foundation models with language-model-as-an-examiner. *Advances in NeurIPS* 36 (2024).
- [4] 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* (2023).
- [5] Ricardo Campos, Vitor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. 2020. YAKE! Keyword extraction from single documents using multiple local features. *Information Sciences* 509 (2020), 257–289.
- [6] Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2024. ChatEval: Towards Better LLM-based Evaluators through Multi-Agent Debate. In *ICLR*.
- [7] Xinyi Chen, Baohao Liao, Jirui Qi, Panagiotis Eustratiadis, Christof Monz, Arianna Bisazza, and Maarten de Rijke. 2024. The sifo benchmark: Investigating the sequential instruction following ability of large language models. *arXiv preprint arXiv:2406.19999* (2024).
- [8] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-Play Fine-Tuning Converts Weak Language Models to Strong Language Models. In *ICML*.
- [9] Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E Gonzalez, et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. See <https://vicuna.lmsys.org> (accessed 14 April 2023) 2, 3 (2023), 6.
- [10] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? Try arc, the ai2 reasoning challenge. *ArXiv preprint* (2018). <https://arxiv.org/abs/1803.05457>
- [11] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training Verifiers to Solve Math Word Problems. *arXiv:2110.14168* [cs.LG]
- [12] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377* (2023).
- [13] 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* (2024).
- [14] 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* (2024).
- [15] Shahul Es. 2023. Orca-chat: A high-quality explanation-style chat dataset.
- [16] Víctor Gallego. 2024. Refined Direct Preference Optimization with Synthetic Data for Behavioral Alignment of LLMs. *arXiv preprint arXiv:2402.08005* (2024).
- [17] Olga Golovneva, Zeyuan Allen-Zhu, Jason Weston, and Sainbayar Sukhbaatar. 2024. Reverse training to nurse the reversal curse. *arXiv preprint arXiv:2403.13799* (2024).
- [18] Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024. From Complex to Simple: Enhancing Multi-Constraint Complex Instruction Following Ability of Large Language Models. *arXiv:2404.15846* [cs.CL]
- [19] Joey Hejna and Dorsa Sadigh. 2024. Inverse preference learning: Preference-based rl without a reward function. *Advances in Neural Information Processing Systems* 36 (2024).
- [20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring Massive Multitask Language Understanding. In *Proceedings of ICLR*. <https://openreview.net/forum?id=d7Kbjm13GmQ>
- [21] Cong Duy Vu Hoang, Philipp Koehn, Gholamreza Haffari, and Trevor Cohn. 2018. Iterative back-translation for neural machine translation. In *2nd Workshop on Neural Machine Translation and Generation*. ACL, 18–24.
- [22] Hamish Ivison, Yizhong Wang, Valentina Pyatkin, Nathan Lambert, Matthew Peters, Pradeep Dasigi, Joel Jang, David Wadden, Noah A Smith, Iz Beltagy, et al. 2023. Camels in a changing climate: Enhancing Lm adaptation with tulu 2. *arXiv preprint arXiv:2311.10702* (2023).
- [23] 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. 2023. Mistral 7B. *arXiv preprint arXiv:2310.06825* (2023).
- [24] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models. In *Proceedings of ACL*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). 4667–4688. <https://aclanthology.org/2024.acl-long.257>
- [25] Abdullatif Köksal, Timo Schick, Anna Korhonen, and Hinrich Schütze. 2023. LongForm: Effective Instruction Tuning with Reverse Instructions. *arXiv preprint arXiv:2304.08460* (2023).
- [26] Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi Rui Tam, Keith Stevens, Abdullah Barhoum, Duc Nguyen, Oliver Stanley, Richard Nagyfi, et al. 2024. Openassistant conversations-democratizing large language model alignment. *Advances in NeurIPS* 36 (2024).
- [27] Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2023. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259* (2023).
- [28] Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024. Self-Alignment with Instruction Back-translation. In *ICLR*.
- [29] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. AlpacaEval: An Automatic Evaluator of Instruction-following Models. https://github.com/tatsubal/alpaca_eval.
- [30] Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*. 74–81.
- [31] Yantao Liu, Zhao Zhang, Zijun Yao, Shulin Cao, Lei Hou, and Juanzi Li. 2024. Aligning Teacher with Student Preferences for Tailored Training Data Generation. *arXiv preprint arXiv:2406.19227* (2024).
- [32] Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. *arXiv preprint cs/0205028* (2002).
- [33] Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of ACL*. 881–893.
- [34] Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *arXiv preprint arXiv:2405.14734* (2024).
- [35] Subhabrata Mukherjee, Arindam Mitra, Ganesh Jawahar, Sahaj Agarwal, Hamid Palangi, and Ahmed Awadallah. 2023. Orca: Progressive learning from complex explanation traces of gpt-4. *arXiv preprint arXiv:2306.02707* (2023).
- [36] OpenAI. 2022. Introducing ChatGPT. <https://openai.com/blog/chatgpt>
- [37] OpenAI. 2024. GPT-4 Technical Report. *arXiv:2303.08774* [cs.CL] <https://arxiv.org/abs/2303.08774>
- [38] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in NeurIPS* 35 (2022), 27730–27744.
- [39] Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction Tuning with GPT-4. *arXiv preprint arXiv:2304.03277* (2023).
- [40] Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. 2024. Pre-training Distillation for Large Language Models: A Design Space Exploration. *ArXiv abs/2410.16215* (2024). <https://api.semanticscholar.org/CorpusID:273507110>
- [41] Chau Minh Pham, Simeng Sun, and Mohit Iyyer. 2024. Suri: Multi-constraint instruction following for long-form text generation. *arXiv preprint arXiv:2406.19371* (2024).
- [42] Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style transfer through back-translation. *arXiv preprint arXiv:1804.09000* (2018).
- [43] Yunjia Qi, Hao Peng, Xiaozhi Wang, Bin Xu, Lei Hou, and Juanzi Li. 2024. ADELIE: Aligning Large Language Models on Information Extraction. *arXiv preprint arXiv:2405.05008* (2024).
- [44] 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* (2024).
- [45] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct Preference Optimization: Your Language Model is Secretly a Reward Model. In *Advances in NeurIPS*, Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (Eds.). http://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html
- [46] Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. 2019. Social IQa: Commonsense Reasoning about Social Interactions. In *Proceedings of*

- EMNLP*. 4463–4473. <https://doi.org/10.18653/v1/D19-1454>
- [47] Rico Sennrich. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* (2015).
- [48] 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* (2024). <https://arxiv.org/abs/2404.02823>
- [49] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models*. 3, 6 (2023), 7.
- [50] Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118* (2024).
- [51] Martina Toshevskaya and Sonja Gievska. 2021. A review of text style transfer using deep learning. *IEEE Transactions on Artificial Intelligence* 3, 5 (2021), 669–684.
- [52] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct Distillation of LM Alignment. *arXiv:2310.16944* [cs.LG]
- [53] Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2022. Finetuned Language Models are Zero-Shot Learners. In *Proceedings of ICLR*. <https://openreview.net/forum?id=gEZrGCzdzqR>
- [54] John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. *arXiv preprint arXiv:1706.01847* (2017).
- [55] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface’s transformers: State-of-the-art natural language processing. *arXiv. arXiv preprint arXiv:1910.03771* (2019).
- [56] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhao Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244* (2023).
- [57] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. 2024. Qwen2 technical report. *arXiv preprint arXiv:2407.10671* (2024).
- [58] Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason E Weston. 2024. Self-Rewarding Language Models. In *ICML*.
- [59] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. HellaSwag: Can a Machine Really Finish Your Sentence?. In *Proceedings of ACL*. 4791–4800. <https://doi.org/10.18653/v1/P19-1472>
- [60] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, et al. 2023. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792* (2023).
- [61] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).
- [62] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. 2023. Instruction-Following Evaluation for Large Language Models. *arXiv preprint arXiv:2311.07911* (2023).
- [63] Zhanhui Zhou, Jie Liu, Jing Shao, Xiangyu Yue, Chao Yang, Wanli Ouyang, and Yu Qiao. 2024. Beyond one-preference-fits-all alignment: Multi-objective direct preference optimization. In *Findings of ACL*. 10586–10613.