Improving the robustness of large language models via consistency alignment

Zhao, Y.; Yan, L.; Sun, W.; Xing, G.; Wang, S.; Meng, C.; ... ; Yin. D.

Improving the Robustness of Large Language Models via Consistency Alignment

Yukun Zhao[1,2], Lingyong Yan[2], Weiwei Sun[1], Guoliang Xing[2], Chong Meng[2]
Shuaiqiang Wang[2], Zhicong Cheng[2], Zhaochun Ren[3*], Dawei Yin[2*]
[1]Shandong University, Qingdao, China, [2]Baidu Inc., Beijing, China
[3]Leiden University, Leiden, The Netherlands
{zhaoyukun02,yanlingyong}@baidu.com,sunnweiwei@gmail.com
{xingguoliang,mengchong01,wangshuaiqiang,chengzhicong01}@baidu.com
z.ren@liacs.leidenuniv.nl,yindawei@acm.org

Abstract

Large language models (LLMs) have shown tremendous success in following user instructions and generating helpful responses. Nevertheless, their robustness is still far from optimal, as they may generate significantly inconsistent responses due to minor changes in the verbalized instructions. Recent literature has explored this inconsistency issue, highlighting the importance of continued improvement in the robustness of response generation. However, systematic analysis and solutions are still lacking. In this paper, we quantitatively define the inconsistency problem and propose a two-stage training framework consisting of instruction-augmented supervised fine-tuning and consistency alignment training. The first stage helps a model generalize on following instructions via similar instruction augmentations. In the second stage, we improve the diversity and help the model understand which responses are more aligned with human expectations by differentiating subtle differences in similar responses. The training process is accomplished by self-rewards inferred from the trained model at the first stage without referring to external human preference resources. We conduct extensive experiments on recent publicly available LLMs on instruction-following tasks and demonstrate the effectiveness of our training framework.

Keywords: Large Language Model, Robustness, Consistency Alignment

1. Introduction

Large language models(LLMs) are now regarded as one of the most advancing fields in artificial intelligence researches (OpenAI, 2023; Chiang et al., 2023; Taori et al., 2023; Touvron et al., 2023). By sufficiently pre-training on massive textual corpus, and carefully fine-tuning and aligning on high-quality instruction-following data, LLMs have demonstrated remarkable capabilities, e.g. understanding human instructions and generating helpful responses (Wei et al., 2021; Ouyang et al., 2022; Dong et al., 2023; Rafailov et al., 2023; Yuan et al., 2023).

However, the robustness of current LLMs, even those leading ones, is still far from promising in recent literature (Gu et al., 2022; Sun et al., 2023; Liang et al., 2023). A commonly observed phenomenon is the inconsistency problem when they respond to distinct but semantically equivalent instructions. We list an example shown in Figure 1: we see GPT-4 returns inconsistent answers to the same task "the referent of the number". Such inconsistency problem reflects the inherent flaws of LLMs to some extent and hinders their practical applications.

Recent work explores the inconsistency problem (Gu et al., 2022; Sun et al., 2023; Li et al., 2023b; Liang et al., 2023). For instance, they find

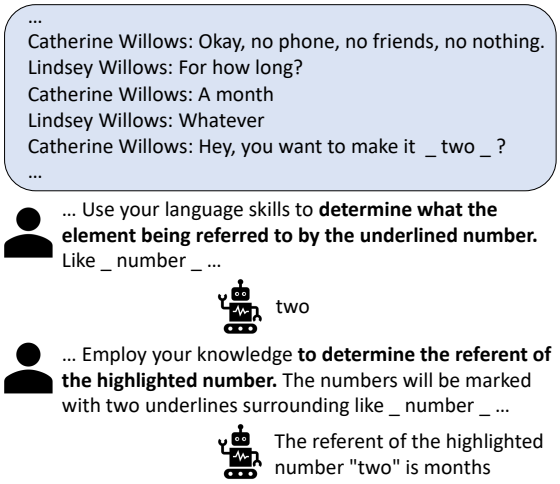


Figure 1: GPT-4 generates inconsistent responses for the identical task.

the LLMs may generate inconsistent responses due to the different verbalized instructions (Li et al., 2023b), data distribution shift (Li et al., 2023a), or even discrepancies in instruction formats (Gu et al., 2022). Based on these observations, Li et al. (2023a) and Liang et al. (2023) propose to optimize the instruction to identify the optimal task instruction that elicits the best performance for LLMs. Nevertheless, there is an absence of quantitative analysis of the current state, along with a systemic solution to improve the instruction-tuned LLMs.

* Co-corresponding authors.

LREC-COLING 2024, pages 8931–8941

In this paper, we first quantitatively analyze the generation robustness of current LLMs in terms of our consistency metrics. We then propose a novel training framework for LLMs via consistency alignment to mitigate the inconsistency problem in current LLMs. Concretely, our training framework sequentially performs the following two training stages: instruction augmented supervised fine-tuning and response consistency alignment. (1) In the augmented supervised fine-tuning (SFT) stage, we first paraphrase the original instruction in the SFT dataset and then pair each paraphrased instruction with the original response to form a new augmented training sample. Finally, all augmented training samples are then added to the SFT dataset to fine-tune the LLMs. (2) In the consistency alignment stage, we feed the paraphrased instructions to LLMs to generate candidate responses, and then construct ‹good, bad› response pairs where each response is individually evaluated by the consistency score. Finally, we optimize the LLMs to directly learn the preferences through an offline training algorithm (Yuan et al., 2023).

We conduct extensive experiments on publicly available models including Vicuna-7B, Vicuna-13B, Llama2-7B, and Llama2-13B on the instruction-following tasks. The experimental results show that by explicitly adding consistency self-alignment, these LLMs can obtain robustness improvements and generalize better on following instructions.

Our contributions are as follows:

1. We propose an integrated training framework to enhance the robustness of LLMs.
2. We propose to utilize self-rewards to improve the performance of a large language model without referring to external human preference resources or external reward models.
3. We conduct extensive experiments to verify the effectiveness of our training framework method across several public LLMs.

2. Related Work

Instruction Tuning. In order to help LLMs understand the instructions and generate human expected responses, recent work (Ouyang et al., 2022; Chiang et al., 2023; Taori et al., 2023; Wei et al., 2021) employ instruction fine-tuning on the pre-trained models to help them follow user instructions. Ouyang et al. (2022) propose to optimize the fine-tuned model (policy model) with PPO to learn human preference. Dong et al. (2023) propose a reward-ranked fine-tuning method, which selects the top n model outputs using an existing reward model to fine-tune foundational LLMs. Rafailov et al. (2023) proposes to directly optimize preferences between two responses given a specific instruction, which implicitly optimizes the same objective as existing PPO algorithms. Song et al. (2023) and Ziegler et al. (2019) propose similar methods to further fine-tune the LLMs utilizing the ranked response pairs that align with human preference. Zhao et al. (2023b) and Zhao et al. (2023a) calibrate the sequence likelihood by sampling generated candidates and making the candidates align with the references in latent space using various ranking loss. Yuan et al. (2023) continue to optimize a bigger model LLAMA-7B and propose RRHF which is a similar method as described above.

Prompting. Prompting is attractive for its simplicity to improve alignment for the LLMs by using few samples or suitable instructions (Brown et al., 2020; Jiang et al., 2021). Wei et al. (2022) propose Chain-of-thought (CoT) to improve reasoning abilities. Their successors (Zhou et al., 2022) propose least-to-most prompting to solve complex reasoning tasks. Wang et al. (2023) and Si et al. (2022) propose to utilize the self-consistency between the sampled answers and choose the most frequent one as the final answer.

Instruction Data. An intuitive start point is to collect a substantial array of diverse and heterogeneous NLP tasks from existing benchmarks (Wei et al., 2021; Longpre et al., 2023; Wang et al., 2022) for instruction-tuning. Then Conover et al. (2023), Köpf et al. (2023) and Chiang et al. (2023) collect crowd-sourcing human-written instructions. Wang et al. (2023), Yu et al. (2023) and Xu et al. (2023) prompt LLMs to generate large-scale, diverse and more complex instructions automatically. Recent work (Zhou et al., 2023; Cao et al., 2023; Chen et al., 2023; Jiang et al., 2023) focus on generating or selecting high-quality and representative instructions to improve the instruction tuning performance.

Robustness on Instruction-following. Recent work (Gu et al., 2022; Liang et al., 2023) have explored that the manipulated instructions would degrade the performance of instruction-tuned LLMs. Li et al. (2023b) evaluate the instruction-following abilities of LLMs through different verbalizations and emphasize the need for continued improvement on instruction-following abilities. Li et al. (2023a) consider the distribution shift between the seen training data and the unseen test data and propose an ensemble method to derive optimal instructions to elicit the performance on the unseen data group. Sun et al. (2023) reveal that instruction-tuned LLMs are sensitive to instruction

re-phrasings, and propose soft prompts by transferring the manipulated instruction to the optimal ones to alleviate this issue. All of the previous work lacks a quantitative analysis of the current state in the robustness of the generations, along with corresponding solutions.

3. Robustness on Instruction Following

Given the multifarious ways in which natural language conveys identical semantics, it is crucial for LLMs to maintain answer consistency across various verbalized questions or instructions. We characterize the robustness in terms of the answer consistency and analyze the current state of LLMs in this regard.

Definition of Consistency We denote Q as a potential space, representing all conceivable linguistic paraphrases conveying equivalent semantic content or user intent. Let $Y : Q \rightarrow \mathcal{P}(Y)$ be a function mapping each question in Q to a probability distribution over the possible responses in Y. Given this, the consistency of an LLM $\mathcal{R}$ is then defined as the expected consistency between the model's responses to any two elements from Q:

$$\mathcal{R} = \mathbb{E}_{q_i,q_j \in Q}\left[\mathbb{E}_{y_i \sim Y(q_i), y_j \sim Y(q_j)}[\text{sim}(y_i, y_j)]\right] \quad (1)$$

where $\text{sim}(y_i, y_j) \in [0, 1]$ is a function measuring the consistency between two responses in Y. A higher value of $\mathcal{R}$ denotes greater robustness. It demonstrate the model's ability to maintain consistency across diverse and potentially infinite linguistic representations in Q, despite the inherent response variability.

For the similarity function, sim, the feasible approaches include measuring the similarity of the response embeddings (Zhang et al., 2020; Pillutla et al., 2021), or exploiting LLMs to check whether the two answers are the same or contradicted (Kadavath et al., 2022). Due to the limitations of semantic similarity based on embeddings and the ability of LLMs that align with human intent better in such tasks, we choose to use LLMs to assess the consistency of responses. That is, we prompt the LLMs to determine whether two responses are similar or contradicted and the instruction is shown in Table 1. Then $sim(y_i, y_j) \in \{0, 1\}$ is obtained by inferring the generated contents.

In particular, we formally define Consistency Rate CR and Maximum Consistency Rate MCR as the consistency metrics. The first one is an indicator of the consistency rate between any two answers under Q,

$$CR = \frac{1}{|Q|} \sum_{Q_k \in Q} \sum_{y_i \in Y_k} \sum_{y_j \in Y_k, j \neq i} \frac{sim(y_i, y_j)}{\binom{|Y_k|}{2}} \quad (2)$$

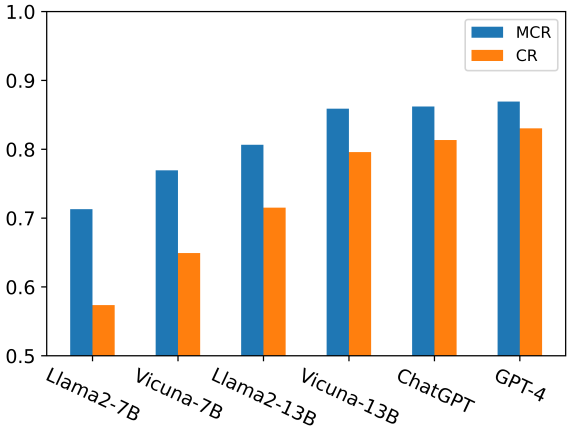


Figure 2: The consistency metrics of recent LLMs.

where $\binom{|Y_k|}{2}$ is the number of 2-combinations for the responses Y_k under the same input. We use Maximum Consistency Rate MCR to report the rate of the maximum consistent answers among the all generated answers on the Q Tasks.

$$MCR = \frac{1}{|Q|} \sum_{Q_k \in Q} \frac{|\Omega_k^{max}|}{|Y_k|} \quad (3)$$

where $|\Omega_k^{max}| = \arg_j |\max \Omega_j|$ and Ω_j is a cluster of consistent answers under the same input.

```
Determine whether answer "A" is the
same or contradicted with the answer
"A Reference" for the question "Q".
For the tasks with fixed answers,
if the two answers are exactly the
same you give "same", otherwise, you
give "Contradicted" as the output.
For free-form generation tasks, you
need to check whether the answer
"A" is an expected generated title,
question, data-to-text description,
or summarization, etc., as the
answer "A Reference". If the two
answers describe a similar meaning
you give "Same", otherwise, you give
"Contradicted" as the output.
```

Table 1: The instruction for determining whether two responses are consistent.

Robustness of the current LLMs We conduct a preliminary study analyzing the current state of contemporary LLMs quantitatively, namely GPT-3.5, GPT-4, Vicuna and LLaMA-2 in terms of the previously defined metrics.

We crafted a test set randomly sampled 490 questions from Super Natural Instructions (Wang et al., 2022), each with 10 different linguistic paraphrases, resulting in a total of 4900 unique questions. These questions spanned diverse topics, including science, literature, mathematics, and gen-

eral knowledge. In our setting, We use GPT-4 to verify the consistency among any two responses. We report the CR and MCR in the Figure 2. We see GPT-4, ChatGPT and Vicuna-13B emerged as the more robust model in terms of the two consistency metrics. There is still room to improve the robustness especially the smaller ones when contending with diverse linguistic representations. The characteristics of the inconsistency are discussed in the following section. These two metrics are straightforward and can be used to illustrate how far the current LLMs are from the optimal robustness performance.

4. Training Large Language Models via Consistency Alignment

We aim to improve the robustness of instruction-following for the large language models via consistency alignment. As shown in Figure 3, our training framework consists of (1) Supervised fine-tuning with instruction augmentation (SFT (IA)) to improve the model's generalization on following instructions; (2) Consistency alignment training (CAT) with the automatic feedback after the first stage, which helps the model notice diversity and subtle differences between the similar responses rather than simple imitation.

4.1. Instruction Augmented Supervised Fine-Tuning

Firstly, we augment the task instructions with similar ones to guide the model's instruction tuning.

Instruction Augmentation Similar instructions are the instructions that convey the same task but are verbalized differently. This aligns with real scenarios where the same task is likely to be induced by different end-users with varying textual expressions. Unfortunately, there is an absence of scaled human-written similar instruction datasets. We prompt the large language models to paraphrase the original instructions into several rephrasings. The language models are not restricted here where can be Vicuna, ChatGPT or GPT-4[1]. In our paper, there is an original task instruction a along with several input-output instances $M = \{(x_i, y_i)\}$ for the task. We paraphrase the task instruction a and keep the input and output instances unchanged. The prompt we use for the paraphrasing task is shown in Table 2.

After the paraphrasing process, we obtain at most n similar task instructions for each task.

[1]We have examined the precision of paraphrasing performance for Vicuna-7B, ChatGPT and GPT-4, the precision values are 93 %, 95% and 95% respectively.

```
Paraphrase the input sentences to
have different words and expressions
but have the same meaning as the
original sentences. Output the various
paraphrases separated by '<br>'.
Please note that you should not answer
the question, but rather paraphrase
it.
```

Table 2: The instruction for the paraphrasing task.

Supervised Fine-Tuning (SFT) Then we use the paraphrased instructions along with its original instruction to fine-tune our model. For each task, we randomly sample m instances for the supervised fine-tuning stage. The training set we use is $S = \bigcup_k \bigcup_j^n \bigcup_i^m \{a_j^k, x_i^k, y_i^k\}$, where a_j^k is the j_{th} task instruction and (x_i^k, y_i^k) is the i_{th} input-output pair for task k. We combine a task instruction a with an input x_i as a question q_i for the model and use y_i as the target output for training. The training objective in our paper is a standard supervised fine-tuning loss shown below:

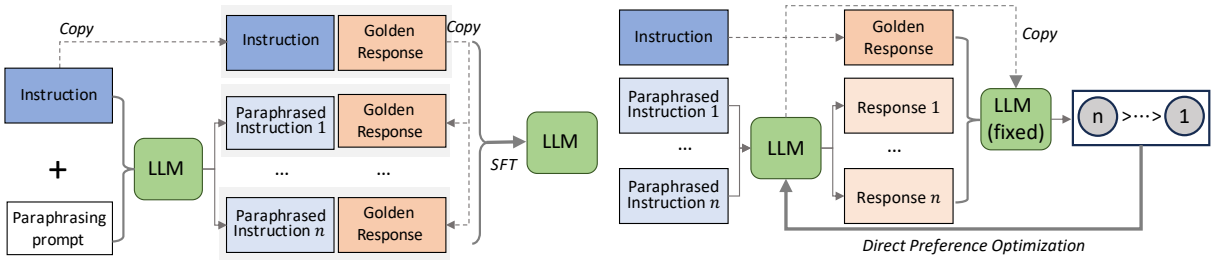
$$L_{sft} = -\sum_t \log P(y_{i,t}|a, x_i, y_{i<t}) \quad (4)$$

4.2. Response Consistency Alignment Training

We obtain a trained model after the first stage. We aren't aiming to train a model that merely mimics and lacks diversity even if its instruction-following capabilities are improved. Therefore, we continue to train the model to learn which responses align with human expectations better, using consistency rewards to differentiate the generated responses. We opt an offline model training method to directly optimize the <good, bad> response pairs for its stability and simplicity like (Rafailov et al., 2023).

For each input x_i, we utilize the trained model to generate n responses in terms of the n task instructions. We build training pairs among the n responses where each response is scored individually via self rewards.

Self Rewards We prompt the trained model to give a reward r_i for each generated response y_i. As we analyze the consistency of the current LLMs in the previous section, we define the inconsistency among the answers in terms of the answer type and the correctness. The answer type indicates whether the model understands the task and is the first step for generation. As shown in Figure 1, the model does not provide a referent but a repeat of the number "two" providing a wrong answer type. The correctness adopts a common definition in language models (Kadavath et al., 2022). We ask the model to output whether the generated



(a) Instruction Augmented Supervised Fine-Tuning (b) Response Consistency Alignment Training

Figure 3: Our consistency alignment training framework.

y_i is the expected answer type with reward $r_i^T \in \{0, 1\}$, and whether the answer is correct with reward $r_i^C \in \{0, 1\}$. The instructions we used to verify the expected answer type and its correctness are shown in the first and second row in Table 3 respectively. We concat the verification instruction along with the question q_i (i.e., $a + x_i$), the generated response y_i and the golden response as the final prompt to obtain the reward from the LLM itself. The r_i^T and r_i^C are then inferred from the generated contents by using the keywords "`Unexpected`", "`Expected`", "`Incorrect`" or "`Correct`". These two tasks are easier for LLMs when used to determine the alignment rewards. We then combine r_i^T and r_i^C to a final reward r_i as follows:

$$r_i = \begin{cases} 0, & r_i^T = 0 \\ 1, & r_i^T = 1 \wedge r_i^C = 0 \\ 2, & r_i^T = 1 \wedge r_i^C = 1 \end{cases} \quad (5)$$

For each input x_i, we obtain response pairs $\bigcup_{j,i} < y_j, y_i >$ where $r_j > r_i \wedge r_j = 2$ through the above step. We then incorporate these preferences into the consistency training process, making the model learn to generate more aligned responses with higher scores, and reduce the probability of generating responses that have lower rewards. Inspired by (Yuan et al., 2023; Rafailov et al., 2023), we optimize this objective by the ranking loss:

$$L_{rank} = \sum_{r_i < r_j} \max(0, p_i - p_j) \quad (6)$$

where p_i is the conditional log probability of y_i under the model. We also add the cross-entropy loss L_{sft}, where the ground-truth y_i here is replaced with the response that have the highest reward $i^{'} = \arg\max_i r_i$. The final loss is the sum of the two losses.

$$L = L_{rank} + \lambda * L_{sft} \quad (7)$$

where λ is the weight of training the model for imitation.

5. Experiments

5.1. Experimental Setup

Dataset We use the Super Natural Instructions (Wang et al., 2022) as our experimental dataset, which consists of 1600+ diverse NLP tasks each with at least one expert-written instruction along with 3155 average input-output samples. We adopt the original division for the training and testing sets and randomly split the original training set into training and validation sets. Finally, we obtain 700, 128, and 56 tasks for training, validation, and testing. For each task, we sample at most 100 instances for training or testing after augmenting the paraphrases.

For testing, we build one more test set to report the consitency metrics. We construct the test set I by utilizing GPT-4 to paraphrase the original task instruction obtaining 10 task instructions, and randomly sampling 10 instances for each task. The test set II is the original 56 tasks along with 100 randomly selected samples per task.

Models We train Vicuna-7B, Vicuna-13B, LLama 2-7B and Llama 2-13B to verify the effectiveness of our proposed training method.

Baselines We compare our training method with the original LLM, the standard supervised fine-tuning (SFT) method, and the off-the-shelf SOTA LLMs including ChatGPT and GPT-4. For standard SFT, we randomly sample 100 instances for each task thus obtaining 70000 total samples for training. We set learning rate 2e-5, epochs 3, and other hyperparameters as the FastChat [2] suggested.

Evaluation Metrics We evaluate our method with robustness and accuracy metrics: (1) Consistency Rate CR of any two answers between the same task Q_k under the test set I; (2) Maximum Consistency Rate MCR reports the rate of

[2] https://github.com/lm-sys/FastChat

`Determine whether the answer "A" is the expected answer type for the question "Q". For tasks with fixed golden answers (like sentiment analysis, entailment inference), you need to check whether the answer is the exact one of the expected enums (like "True", "False", "Positive", "Negative", "A", "B", "Contradiction", "Neutral" or "Entailment".) mentioned in the question "Q". For free-form generation tasks (like title generation, question generation, data-to-text generation), you need to check whether the answer is an expected generated title, question, data-to-text description or summarization, etc., as the question "Q" required. You also need to compare with the "Golden A" to determine whether the answer type aligns with the answer type of the "Golden A". If the answer has the same type as the golden answer, give "Expected answer type", otherwise give "Unexpected answer type". Please note that you only need to determine whether it matches the instructed answer type, and do not need to verify whether the answer is correct.`
`Determine whether the answer "A" is "Correct" or "Incorrect" for question "Q". For tasks with fixed golden answers (the answer is limited to a finite set such as "True", "False", "Positive", "Negative", "A", "B", "Contradiction", "Neutral", "Entailment"), you need to check whether the answer exactly matches (equals) the golden answer "Golden A". For free-form generation tasks (the answer is a free-form generation and not unique such as title, question, data-to-text description generation or summarization), you need to check whether the answer describe the same thing as the golden answer, or the answer is fluent, plausible for the question "Q". If the answer is correct, give "Correct", otherwise give "Incorrect" as the result.`

Table 3: The instructions for determining whether a response is the expected answer type and whether it is a correct answer.

the maximum consistent answers under the test set I; (3) ROUGE-1, ROUGE-L under the test set I and test set uppercaseii . Besides, we perform the human evaluation by annotating the quality of the generated responses with scores 0, 1, and 2 ourselves. We report the number of wins, ties, and loses across the responses generated by different models.

Implementation Details For instruction augmentation, we utilize Vicuna-7B, Vicuna-13B, and ChatGPT to paraphrase the original task instructions obtaining at most 30 instructions. We randomly sample $n = 10$ instructions and sample 10 instances for each task for the subsequent training. Thus, the scale of training resources is the same as the baseline. For the SFT stage, we train the models using the FastChat with ZeRO-3 on 4 80G A100 GPUs, setting training epochs 3 and the other hyperparameters as the FastChat suggested. For the CAT stage, we revise minor codes in DPO in LLaMA-Efficient[3] to support our training objective. The rewards are inferred from the LLMs themselves for training, i.e., Vicuna-7B, Vicuna-13B, Llama 2-7B, and Llama 2-13B respectively. We train the models with 3 epochs using LoRA (Hu et al., 2021) and ZeRO-3 on 4 40G A100 GPUs and set $\lambda = 1$, $lora\_r = 8, lora\_target = q\_proj, k\_proj, v\_proj, o\_proj$ and $lr = 1e-5$. The other hyperparameters are the default setting. To fairly compare the effectiveness of training methods, we do not import any additional data in this stage and re-use the training samples at the SFT stage.

[3]https://github.com/hiyouga/LLaMA-Efficient-Tuning

5.2. Main Results

We report the consistency metrics CR, MCR[4] as well as ROUGE-1 and ROUGE-L on the compared models and training methods on test set I in Table 4. We observe a significant improvement in the consistency scores and ROUGE scores for the Vicuna and Llama 2 models after SFT. SFT with Instruction Augmentation (SFT(IA)) results in higher consistency scores and ROUGE scores compared to standard SFT. The ROUGE scores are improved as well which indicates the instruction augmentation can help to generalize on following instructions. Building upon the SFT (IA) model, consistency alignment training (CAT) brings continual improvements in consistency scores and ROUGE scores for all the compared Vicuna and LLama2 models. Vicuna-13B + SFT (IA) + CAT surpasses the SOTA LLM GPT-4 in our setting. These results demonstrate the effectiveness of our training method (SFT (IA) + CAT).

Besides, we observe that the performance obtained after SFT (IA) and CAT based on Vicuna is

[4]We prompt GPT-4 to judge whether two answers are consistent in the testing stage for consistency metrics, and the prompts are those we discussed in section 3.

	CR	MCR	ROUGE-1	ROUGE-L
GPT-4	0.8303	0.8693	0.3870	0.3751
ChatGPT	0.8134	0.8620	0.3022	0.2744
Vicuna-7B	0.6492	0.7694	0.1385	0.1266
Vicuna-7B + SFT	0.7092	0.8123	0.3782	0.3672
Vicuna-7B + SFT (IA)	0.7753	0.8504	0.3894	0.3757
Vicuna-7B + SFT (IA) + CAT	0.8298	0.8743	0.4187	0.4097
Vicuna-13B	0.7959	0.8589	0.1724	0.1596
Vicuna-13B + SFT	0.8017	0.8490	0.4028	0.3903
Vicuna-13B + SFT (IA)	0.8267	0.8619	0.4131	0.4014
Vicuna-13B + SFT (IA) + CAT	0.8390	0.8804	0.4276	0.4185
Llama2-7B	0.5735	0.7129	0.0637	0.0492
Llama2-7B + SFT	0.7702	0.8308	0.2682	0.2560
Llama2-7B + SFT (IA)	0.7921	0.8475	0.2901	0.2733
Llama2-7B + SFT (IA) + CAT	0.8107	0.8521	0.3012	0.2806
Llama2-13B	0.7151	0.8065	0.0737	0.0627
Llama2-13B + SFT	0.7505	0.8180	0.3085	0.2975
Llama2-13B + SFT (IA)	0.7589	0.8282	0.3379	0.3280
Llama2-13B + SFT (IA) + CAT	0.8100	0.8601	0.3711	0.3502

Table 4: The overall performance of compared methods and models on the test set I.

	ROUGE-1	ROUGE-L
GPT-4	0.4506	0.4408
ChatGPT	0.3187	0.3051
Vicuna-7B	0.1702	0.1570
+SFT	0.4085	0.3929
+SFT (IA)	0.4122	0.3984
+SFT (IA) + CAT	0.4391	0.4285
Vicuna-13B	0.2102	0.1972
+SFT	0.4234	0.4071
+SFT (IA)	0.4477	0.4350
+SFT (IA) + CAT	0.4683	0.4417
Llama2-7B	0.0684	0.0513
+SFT	0.2743	0.2614
+SFT (IA)	0.3163	0.2903
+SFT (IA) + CAT	0.3189	0.2977
Llama2-7B	0.0745	0.0643
+SFT	0.3351	0.3215
+SFT (IA)	0.3697	0.3587
+SFT (IA) + CAT	0.4289	0.4066

Table 5: The ROUGE scores of the compared models and methods on the test set II.

superior to that based on Llama 2, confirming the importance of choosing the base model for further training.

We report the ROUGE-1 and ROUGE-L scores on the standard test set II shown in Table 5. The improvements and conclusions across the compared methods are consistent as those observed in test set I. Since the Vicuna-13B achieves the best performance, we use it as the backbone for detailed analysis in the next section.

Rewards	ROUGE-1	ROUGE-L
r^C from SFT	0.4123	0.4051
$r^C + r^T$ from SFT	0.4276	0.4185
$r^C + r^T$ from Vicuna-13B	0.3962	0.3877

Table 6: The ROUGE scores on test set I when the rewards are expected type or expected type + answer correctness from Supervised fine-tuned Vicuna-13B and vanilla Vicuna-13B.

5.3. Detailed Analysis

The Choice of Rewards We study whether it is necessary to split the rewards into two steps: one for determining whether it is the expected answer type and the next for correctness. We report the ROUGE values for vicuna-13B on test set I when we construct alignment training pairs using only the correctness reward r^C or a combination of correctness and answer type rewards $r^C + r^T$ in Table 6. We observe that the ROUGE values have been improved when combining the rewards together, as determining whether it is the expected answer type is an easier task which serves as an auxiliary task to enhance the accuracy of the self rewards. Besides, we study the impact of rewards from different models, i.e., the vanilla Vicuna-13B and the fine-tuned Vicuna-13B with instruction augmentations. We see that using the rewards from a less aligned model would degrade the CAT performance and its ROUGE values are even lower than its SFT version shown in Table 5.

The Choice of λ We compare the performance of different coefficients of the SFT loss in the final training objective. We report the ROUGE-1 and ROUGE-L scores for Vicuna-13B on test set

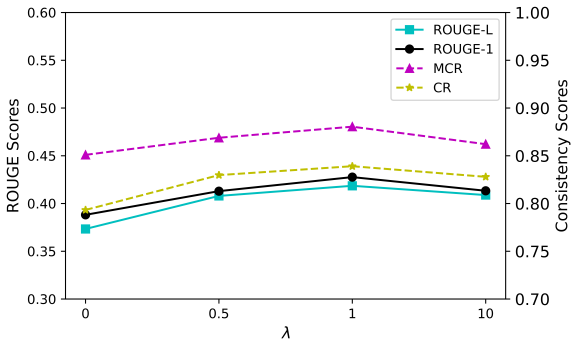


Figure 4: The performance of different λ for our training method on the test set I.

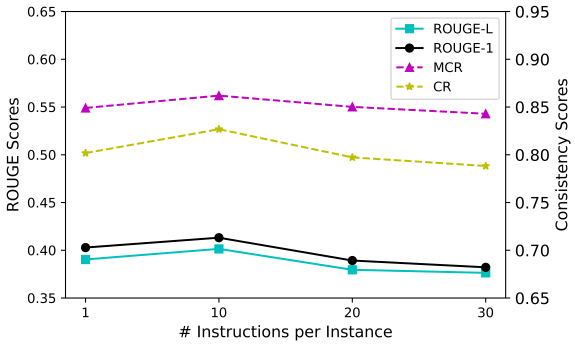


Figure 5: The performance of SFT (IA) across varying number of instructions for each input on the test set I .

I when λ is set to 0, 0.5, 1, and 10 in Figure 4. We notice the importance of adding the SFT loss to the CAT objective as the ROUGE scores are higher when we set λ to 1 or 0.5 compared with 0. When we further increase λ, we make the model learn less from negative generations and reduce the benefits of consistency alignment training. This would decrease the performance in terms of the ROUGE scores.

The Number of Augmented Instructions We further study whether the performance would be improved when we increase the number of augmented instructions. For fairness, we keep the size of the training set and other hyperparameters fixed and only increase the number of augmented instructions. We report the metrics of the fine-tuned Vicuna-13B with 1, 10, 20, and 30 instructions per task on test set I in Figure 5. We see that when we increase the number of instructions to 10, both the generated consistency and ROUGE metrics are improved. However, continually increasing the number of instructions would degrade the performance as the number of training instances for each task decreases accordingly. This encourages that we need to ensure each task is sufficiently trained instead of consistently increasing the number of instructions.

Strategy	Baseline	diff.	win	tie	lose
CAT+SFT(IA)	Vanilla	83	48	44	8
CAT+SFT(IA)	SFT	46	32	55	13
CAT+SFT(IA)	SFT(IA)	27	31	60	9

Table 7: Human evaluation on test set I. We report the different ratio, the number of wins, ties and loses among the responses generated by the strategy and baseline models. All the models are trained based on Vicuna-13B and Vanilla denotes the vanilla Vicuna-13B.

5.4. Human Evaluation

We perform human evaluation on the generated responses across different trained models. To compare any two trained models, we sample the different responses (not exactly match, abbreviated as "diff.") generated by the two models in test set I, and we then manually evaluate the pair of these responses[5]. Based on the statistics in Table 7, the performance has been significantly improved through CAT + SFT (IA) compared with its vanilla version, with more wins than loses and a greater diff. ratio. We analyze the human-labeled scores when reporting wins or loses to gain a deeper understanding of how the wins happen. The ratios of responses scored 1 or 2 and 2 are 86% and 44% respectively for CAT + SFT (IA), and the ratios are only 56% and 14% for vanilla, verifying the model has been trained to generate more expected answer types and more correct answers. When comparing CAT + SFT (IA) with SFT, the gap between wins and loses indicates the superior performance of CAT + SFT (IA). We see that the model trained with consistency alignment after SFT (IA) can obtain consistent improvements compared with SFT (IA) in terms of more wins than loses. This indicates that the model can be continually improved to generate more expected answers with additional CAT.

From all the above human evaluated results, we demonstrate that CAT + SFT(IA) training method helps to generate more aligned responses than the other trained methods.

6. Conclusion

In this paper, we investigate the robustness of current large language models in terms of the consistency of the generated responses. We introduce a novel training framework to boost the robustness of LLMs including instruction-augmented supervised fine-tuning (SFT (IA)) and response con-

[5]We evaluate whether a response is an expected answer type and its correctness and score 0, 1, 2 as we discussed in section4.2. The number of wins, ties, and loses are inferred by comparing the two scores.

sistency alignment training (CAT). We conduct extensive experiments on Vicuna and Llama 2 on the instruction-following tasks and validate the effectiveness of our training framework. Additionally, we separately verify the effectiveness of the SFT (IA) and CAT modules. The method proposed in this paper serves as a plug-in for continuously improving the performance of existing LLMs. Furthermore, it eliminates the need for additional human guidance or external reward models, which use decomposed self-rewards to help the model generate more robust and accurate responses. We believe this approach can contribute to advancing the research on generation robustness to some extent.

7. limitations

While our consistency alignment training is effective for improving the robustness and generalization of following instructions, it has several limitations. Firstly, our training approach relies on a model's self-rewards, so if the performance of the model's alignment is poor, the rewards we obtain may not lead to further improvements. In the future, we plan to conduct experiments on smaller and weaker large language models. Besides, the diversity of the verbalized instructions may be limited compared with end-users as we collect the re-phrasings through LLMs. We plan to collect instructions from a wide range of end-users to construct test and training datasets to evaluate and improve the robustness of model generation.

8. Ethics Statement

This work does not have explicit ethical considerations as all the models and datasets we used are public. We acknowledge that the LLMs may encode problematic biases. It is unclear how the training process might interact with these problems.

9. Acknowledgements

This work was supported by the National Key R&D Program of China with grant No.2020YFB1406704, the Natural Science Foundation of China (62272274, 61902219, 61972234), the Natural Science Foundation of Shandong Province (ZR2021QF129).

10. Bibliographical References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*, pages 1877–1901.

Yihan Cao, Yanbin Kang, and Lichao Sun. 2023. Instruction mining: High-quality instruction data selection for large language models. *arXiv preprint arXiv:2307.06290*.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. 2023. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*.

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

Mike Conover, Matt Hayes, Ankit Mathur, Xiangrui Meng, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, et al. 2023. Free dolly: Introducing the world's first truly open instruction-tuned llm.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Jiasheng Gu, Hanzi Xu, Liangyu Nie, and Wenpeng Yin. 2022. Robustness of learning from task instructions. *arXiv preprint arXiv:2212.03813*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. In *ICLR*.

Yuxin Jiang, Chunkit Chan, Mingyang Chen, and Wei Wang. 2023. Lion: Adversarial distillation of closed-source large language model. *arXiv preprint arXiv:2305.12870*.

Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. 2021. How can we know when language models know? on the calibration of language models for question answering. *TACL*, 9:962–977.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield-Dodds, Nova DasSarma, Eli Tran-Johnson, et al. 2022. Language models (mostly) know what they know. *arXiv preprint arXiv:2207.05221*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. 2023. Openassistant conversations–democratizing large language model alignment. *arXiv preprint arXiv:2304.07327*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *EMNLP*, pages 3045–3059.

Moxin Li, Wenjie Wang, Fuli Feng, Jizhi Zhang, and Tat-Seng Chua. 2023a. Robust instruction optimization for large language models with distribution shifts. *arXiv preprint arXiv:2305.13954*.

Shiyang Li, Jun Yan, Hai Wang, Zheng Tang, Xiang Ren, Vijay Srinivasan, and Hongxia Jin. 2023b. Instruction-following evaluation through verbalizer manipulation. *arXiv preprint arXiv:2307.10558*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *ACL*, pages 4582–4597.

Shihao Liang, Kunlun Zhu, Runchu Tian, Yujia Qin, Huadong Wang, Xin Cong, Zhiyuan Liu, Xiaojiang Liu, and Maosong Sun. 2023. Exploring format consistency for instruction tuning. *arXiv preprint arXiv:2307.15504*.

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

Renze Lou, Kai Zhang, and Wenpeng Yin. 2023. Is prompt all you need? no. a comprehensive and broader view of instruction learning. *arXiv preprint arXiv:2303.10475*.

OpenAI. 2023. Gpt-4 technical report. *ArXiv*, abs/2303.08774.

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. *NIPS*, 35:27730–27744.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. 2021. Mauve: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems*, 34:4816–4828.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*.

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Boyd-Graber, and Lijuan Wang. 2022. Prompting gpt-3 to be reliable. *arXiv preprint arXiv:2210.09150*.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2023. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*.

Jiuding Sun, Chantal Shaib, and Byron C Wallace. 2023. Evaluating the zero-shot robustness of instruction-tuned language models. *arXiv preprint arXiv:2306.11270*.

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):7.

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*.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *ICLR*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *EMNLP*, pages 5085–5109.

Albert Webson and Ellie Pavlick. 2022. Do prompt-based models really understand the meaning of their prompts? In *NAACL*, pages 2300–2344.

Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*, pages 24824–24837.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*.

Yue Yu, Yuchen Zhuang, Jieyu Zhang, Yu Meng, Alexander Ratner, Ranjay Krishna, Jiaming Shen, and Chao Zhang. 2023. Large language model as attributed training data generator: A tale of diversity and bias. *arXiv preprint arXiv:2306.15895*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. In *NeurIPS*.

Qingru Zhang, Minshuo Chen, Alexander Bukharin, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. 2023. Adaptive budget allocation for parameter-efficient fine-tuning. In *ICLR*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *ICLR*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023a. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.

Yao Zhao, Misha Khalman, Rishabh Joshi, Shashi Narayan, Mohammad Saleh, and Peter J Liu. 2023b. Calibrating sequence likelihood improves conditional language generation. In *ICLR*.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. 2023. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. In *ICLR*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.