RNR: Teaching Large Language Models to Follow Roles and Rules

Anonymous ACL submission

Abstract

Instruction fine-tuning (IFT) elicits instruction following capabilities and steers the behavior of large language models (LLMs) via supervised learning. However, existing models trained on open-source IFT datasets only have the ability to follow instructions from users, and often fail to follow complex role and rules specified by developers, a.k.a. system prompts. The ability to follow these roles and rules is essential for deployment, as it ensures that the model safely interacts with users within developer defined guidelines. To bridge this gap, we propose RoleNRules, an automated data generation pipeline that generates diverse roles and rules from existing IFT instructions, along with corresponding responses. This data can then be used to train models that follow complex system prompts. Experimental results show that RoleNRules can significantly improve LLMs' role and rule following capabilities, as evidenced by over 25% increase in pass-rate of following all requirements in system prompts, without regression on popular instruction following benchmarks.

1 Introduction

Instruction fine-tuning (IFT) (Ouyang et al., 2022; Longpre et al., 2023) has emerged as a powerful technique to elicit latent capabilities and steer the behaviors of Large Language Models (LLMs) (Qin et al., 2023; OpenAI, 2023; Chowdhery et al., 2022). The core of IFT lies in the generation and selection of instruction data (Chen et al., 2024; Chung et al., 2022; Lee et al., 2023; Honovich et al., 2022; Wang et al., 2022; Zhou et al., 2023a), which leverage existing LLMs like InstructGPT (Ouyang et al., 2022) and ChatGPT (OpenAI, 2022) to recursively generate instruction sets. By fine-tuning with such carefully curated instruction-response pairs, IFT enables LLMs to generate desirable responses to user instructions (Wang et al., 2022; Taori et al., 2023).

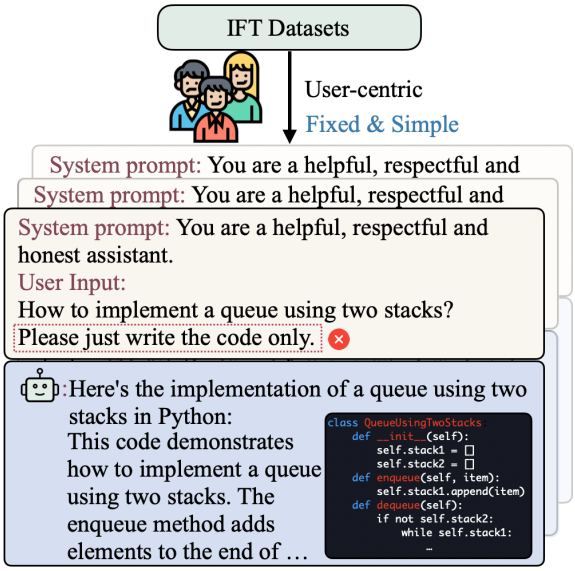

Figure 1: Generated response for a model trained with exsiting IFT dataset, which ignores the user's extra requirement to write the code only.

Current IFT research is mainly focused on handling user input instructions. However, this overlooks the need of developers to control LLM behavior via specific role and rule requirements. This enhanced level of controllability is necessary, as developers often need to specify the LLM's role in their application ("*You are an AI coding assistant*") as well as specific rules ("*Please just write the code only*"). To enable such customization ability, many closed-source LLMs like GPT-4* and Claude † allow developers to input roles and rules in system prompt, which can control the LLMs behavior when interacting with the user. However, these techniques to enable such role and rule following ability are currently not clear to the research community.

Existing open-source IFT datasets commonly

*https://platform.openai.com/docs/guides/prompt-engineering/tactic-ask-the-model-to-adopt-a-persona

†https://docs.anthropic.com/claude/docs/how-to-use-system-prompts

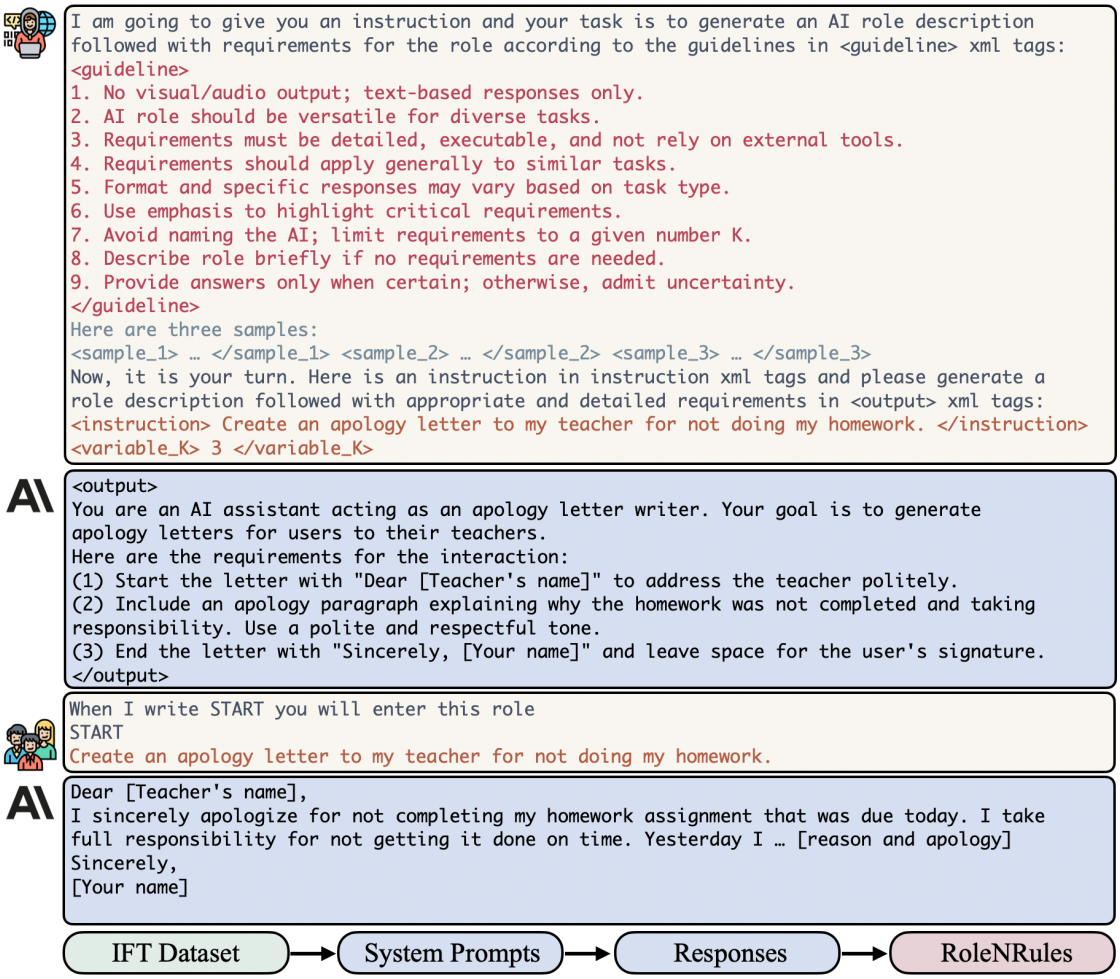

Figure 2: To construct (system prompt, instruction, response) triplets, RNR utilizes existing open-source IFT datasets and a powerful LLM, such as Claude2. First, we crafted a comprehensive guideline, accompanied by a few-shot demonstration. These serve as prompts for the LLM, guiding it to generate informative system prompts for the instructions in the given IFT dataset. Subsequently, we merge the generated system prompts with their corresponding instructions and feed them into the LLM to obtain the responses. Finally, we assemble these components into (system prompt, instruction, response) triplets to form the final dataset.

employ a fixed system prompt (e.g., *"You are a helpful, respectful, and honest assistant."*). One workaround to enable role and rule following ability is to prepend these requirements to user inputs, but we find this approach is not very effective. We argue that it is because open source IFT models are not trained with diverse role and rule system prompts, and developer role and rule requirements are often complex to follow, causing a significant shift between the training and inference. To illustrate this challenge, consider the example in Figure 1, where we ask a model trained on a popular open-source IFT dataset to generate code only, but the system ignores this requirement and includes undesirable information.

To bridge this gap, we propose RoleNRules (RNR), a scalable and automated data generation pipeline designed to enhance the ability of LLMs to follow developer instructions specified in the system prompt, as illustrated in Figure 2. The core of our approach involves augmenting existing IFT datasets by generating tailored system prompts and corresponding responses. Specifically, we leverage Claude-2 to produce system prompts that combine role descriptions and specific operational rules. These prompts reflect diverse developer needs and requirements, based on existing instructions from IFT datasets. We then query Claude-2 again to generate responses by combining the newly generated system prompt with the original instruction, ensuring both role consistency and rule adherence.

These (system prompt, instruction, response) triplets are used for training the model, resulting in superior adaptability to system prompts, as evidenced by over 25% increase in pass-rate of following all requirements in system prompts, without regression on popular instruction following benchmarks.

2 Experiments

2.1 Experimental Setup

Training Setup. We present our main results on the Alpaca dataset (Taori et al., 2023) and the Ultrachat dataset (Ding et al., 2023). For Alpaca, we generate (system prompt, instruction, response) triplets on the whole dataset and for Ultrachat, we sample a subset of size 50K to generate the triplets. For our main experiments, we use Llama 2 7B (Touvron et al., 2023) as our base model and fine-tune each model for two epochs with a batch size of 128 using a context length of 2048 and a learning rate of 2×10^{-5}.

Baselines. To investigate the efficacy of RNR, we compare it with two IFT baselines. For both of these baselines we generate responses to the original training instructions with Claude-2, so that there is no discrepancy in the teacher model used for training. Once we generate these responses, we train models using (a) a fixed system prompt ("You are a helpful, harmless, and honest AI assistant") and (b) with no system prompt. These baselines are denoted as IFT: fixed-system and IFT: no-system, respectively.

2.2 Benchmarks

We design three benchmarks using the RNR format (system prompt + instruction) to evaluate the effectiveness of RoleNRules:

1. **RNR-Expert benchmark**: This is a small benchmark comprised of 50 data points with RNR system prompts and instructions handcrafted by domain experts, who are LLM researchers and are familiar with prompting techniques. The roles and rules in these system prompts cover diverse domains including programming, engineering, medicine, law, finance, movies, book recommendations, travel advice, and cooking recipes, etc. We remark that this dataset contains challenging and high quality examples.

2. **RNR-Awesome benchmark**: This is benchmark has 495 data points based on the instruction collected by awesome-chatgpt-prompts (Akın, 2023). For each instruction in awesome-chatgpt-prompts, we randomly select 6 samples from the RNR-Expert benchmark as the few-shot prompt, then leverage GPT-4 to generate the RoleNRules system prompt for this instruction. Given this system-prompt, we also use GPT-4 to generate 2 more possible instructions, so the amount of data points in RNR/Awesome is 3 times of the number of prompts in the awesome-chatgpt-prompts dataset.

3. **RNR-AlpacaFarm benchmark**: This benchmark contain 805 datapoints, and the instructions are kept the same as the original AlpacaFarm (Dubois et al., 2023). We also use 6 random samples from the RNR-Expert benchmark as the few-show prompt and leverage GPT-4 to generate the RoleNRules system prompt for each instruction in AlpacaFarm.

Evaluation. To evaluate the system-prompt following ability of each model, we use large language models to calculate the number of rules and roles that are satisfied for each prompt. In particular, we develop a novel chain-of-thought (Wei et al., 2022b) like prompt for evaluating RoleNRules following ability. More concretely, we ask the judge LLM to examine each role and rule requirement individually, and then aggregate these score into a final judgement. In addition, we include some few-shot examples to further boost performance. The exact prompt can be found in Appendix J. In our human study, we find that this evaluation strategy has a high level of agreements with an ensemble of human experts. Claude-2.1 and the majority vote of the human annotators agree 73.97 % of the time, and that human evaluators agree 77.32 % of the time. We defer more details into Appendix A.

In our experiments, we use Claude-2.1 to calculate the percentage of rules in the system prompt that each model passes over each RNR benchmark dataset. We remark that we use Claude-2.1 as the evaluator instead of Claude-2 to mitigate the effect of LLM judges favoring their own output. We calculate the rule-level pass percentage (defined as the percent of rules each model passes) and the prompt-level pass percentage (defined as the percent of prompts where all rules are satisfied).

2.3 Main Results

Our main results on the RNR benchmark can be found in Table 1. The results show that RNR can significantly improve system-prompt following ability compared to the two IFT baselines for both the Alpaca and Ultrachat datasets. In particular, we find that RNR improves prompt-level system-prompt following ability by 37.4% on the Alpaca

Table 1: Pass rate on the main evaluation sets. Llama-2-7B is the base model for RNR.

RNR:Alpaca	RNR-Expert		RNR-Awesome		RNR-Alpacafarm		Average	
	Rule Level	Prompt Level	Rule Level	Prompt Level	Rule Level	Prompt Level	Rule Level	Prompt Level
IFT: fixed-system	53.62	39.53	58.39	28.6	72.57	43.21	61.53	37.11
IFT: no-system	64.76	57.78	67.59	39.05	74.41	44.47	68.92	47.10
Rolenrule	**82.24**	**77.78**	**74.91**	**49.07**	**86.34**	**67.30**	**81.16**	**64.72**
RNR:Ultrachat	Rule Level	Prompt Level	Rule Level	Prompt Level	Rule Level	Prompt Level	Rule Level	Prompt Level
IFT: fixed-system	61.18	48.89	57.51	29.10	76.14	49.56	64.94	42.52
IFT: no-system	80.50	70.21	63.07	36.49	75.23	51.09	72.93	52.60
Rolenrule	**85.56**	**75.0**	**75.17**	**49.59**	**89.56**	**74.37**	**83.43**	**66.32**

dataset and by 26.1% for Ultrachat compared to the strongest baseline. These results indicate that RNR is an effective way to improve system-prompt following ability.

Besides RNR benchmark, we also run evaluations on public benchmarks, namely IFEval, Rolebench and Alpacafarm. The results for IFEval can be found in Table 2. From Table 2 we can see that RNR significantly boosts instruction following ability, even when the instruction is inputted by the user, indicating that improve system prompt following capability can positively transferred to user inputs. For RoleBench, we find that RNR improves the winning score (defined as $\frac{\text{\# Win - \# Lose}}{\text{\#Win + \#Lose + \#Tie}} + 1$) by over 25% compared to both IFT baselines on the Alpaca dataset. The results can be found in Table 3. This indicates RNR significantly enhances role-playing ability. For evaluation on Alpacafarm, we do a head to head comparison of RNR:Alpaca and IFT:no-system. We find that the performance is similar for IFT:no-system and RNR, with RNR outperforming the IFT model $\sim 3.7\%$. (See Appendix C). Therefore RNR can boost complex system-prompt following ability without hurting general instruction following performance. We also demonstrate the performance of RNR on various benchmarks in Table 5. As expected, the benchmark performance is similar for all models, which shows that RNR retains general reasoning capabilities.

Table 2: Pass rate on the IFEval. Llama-2-7B is the base model.

Alpaca	IFEval:Loose		IFEval:Strict	
	Rule Level	Prompt Level	Rule Level	Prompt Level
IFT: fixed-system	39.69	27.73	32.25	19.69
IFT: no-system	39.69	26.80	32.01	19.41
RNR	**51.68**	**39.57**	**43.65**	**31.61**
Ultrachat	Rule Level	Prompt Level	Rule Level	Prompt Level
IFT: fixed-system	39.69	27.34	30.70	19.04
IFT: no-system	40.17	28.10	30.70	19.04
RNR	**47.84**	**34.57**	**40.17**	**27.17**

2.4 Varying the Amount of RNR Data

We investigate how the amount of rolenrule data affects rule following ability in Figure 3. From this figure we observe that only a small amount of RNR data (1000 data points) will significantly improve system-prompt following ability compared to the baselines. In addition, increasing the amount of RNR data included further boosts performance, further confirming RNR's efficacy.

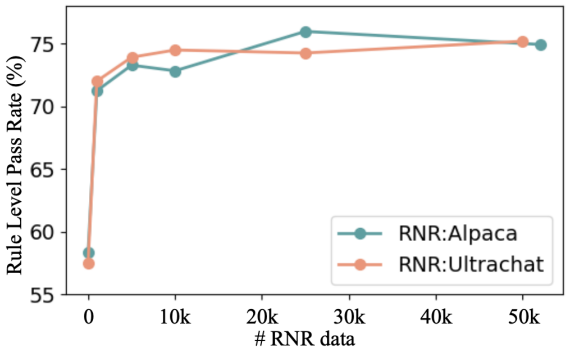

Figure 3: Ablation on the amount of RoleNRules data included. The evaluation dataset is the Awesome split.

2.5 Human Evaluation

To validate the efficacy of RNR, we conducted a human evaluation experiment in addition to using public LLM APIs for response evaluation. In this experiment, expert human researchers evaluated the prompt-level pass rate of both the RNR model and the IFT: no-system model using an expert-curated set from the RNR benchmark. Our results show that RNR has significantly better pass percentage, both when considering strict evaluation (All Pass) and non-strict evaluation (Majority Pass). We defer more details into Appendix A.

3 Conclusion

In conclusion, our study presents a new approach to IFT for LLMs, focusing on both user-centric instructions and developer-defined roles and rules. Our novel system prompt structure and automated data generation pipeline contribute to significant improvements in rule-following capabilities in LLMs, as evidenced by enhanced performance on our benchmarks using the Alpaca and Ultrachat datasets.

4 Limitation

Despite these advancements, our work has certain limitations. First, our data generation pipeline currently lacks a quality-based filtering mechanism, which could affect the overall quality and consistency of the training data. This may result in the model occasionally producing suboptimal responses due to less rigorous data curation.

Additionally, our framework is limited to single-turn interactions, which restricts its applicability to more complex, multi-turn conversation settings. Future efforts will focus on expanding this to multi-turn dialogues to improve the model's effectiveness in sustaining coherent and contextually appropriate conversations over extended interactions.

Furthermore, training models to follow specific roles poses ethical challenges, as there is a risk of reinforcing stereotypes or inadvertently encouraging ethically questionable behaviors. Addressing these issues will be a priority in future work to ensure that role-following behaviors are implemented responsibly and ethically.

References

Fatih Kadir Akın. 2023. awesome-chatgpt-prompts: A curated list of awesome chatgpt prompts. https://github.com/f/awesome-chatgpt-prompts.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024. Alpagasus: Training a better alpaca model with fewer data. In *The Twelfth International Conference on Learning Representations*.

Zhiyu Chen, Shiyang Li, Charese Smiley, Zhiqiang Ma, Sameena Shah, and William Yang Wang. 2022. ConvFinQA: Exploring the chain of numerical reasoning in conversational finance question answering. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6279–6292, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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)*.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpacafarm: A simulation framework for methods that learn from human feedback. *arXiv preprint arXiv:2305.14387*.

Jacob Eisenstein. 2022. Informativeness and invariance: Two perspectives on spurious correlations in natural language. *arXiv preprint arXiv:2204.04487*.

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Jörg Frohberg and Frank Binder. 2021. Crass: A novel data set and benchmark to test counterfactual reasoning of large language models. *arXiv preprint arXiv:2112.11941*.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. 2021. Causal abstractions of neural networks. *Advances in Neural Information Processing Systems*, 34:9574–9586.

Atticus Geiger, Zhengxuan Wu, Hanson Lu, Josh Rozner, Elisa Kreiss, Thomas Icard, Noah Goodman, and Christopher Potts. 2022. Inducing causal structure for interpretable neural networks. In *International Conference on Machine Learning*, pages 7324–7338. PMLR.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022. Unnatural instructions: Tuning language models with (almost) no human labor. *arXiv preprint arXiv:2212.09689*.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2023. Can large language models truly understand prompts? a case study with negated prompts. In *Transfer Learning for Natural Language Processing Workshop*, pages 52–62. PMLR.

Emre Kıcıman, Robert Ness, Amit Sharma, and Chenhao Tan. 2023. Causal reasoning and large language models: Opening a new frontier for causality. *arXiv preprint arXiv:2305.00050*.

Matt J Kusner, Joshua Loftus, Chris Russell, and Ricardo Silva. 2017. Counterfactual fairness. *Advances in neural information processing systems*, 30.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Cheng Li, Ziang Leng, Chenxi Yan, Junyi Shen, Hao Wang, Weishi MI, Yaying Fei, Xiaoyang Feng, Song Yan, HaoSheng Wang, et al. 2023a. Chatharuhi: Reviving anime character in reality via large language model. *arXiv preprint arXiv:2308.09597*.

Shiyang Li, Jianshu Chen, Yelong Shen, Zhiyu Chen, Xinlu Zhang, Zekun Li, Hong Wang, Jing Qian, Baolin Peng, Yi Mao, et al. 2022. Explanations from large language models make small reasoners better. *arXiv preprint arXiv:2210.06726*.

Shiyang Li, Jianshu Chen, and Dian Yu. 2019. Teaching pretrained models with commonsense reasoning: A preliminary kb-based approach. *ArXiv*, abs/1909.09743.

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*.

Bill Yuchen Lin, Yicheng Fu, Karina Yang, Prithviraj Ammanabrolu, Faeze Brahman, Shiyu Huang, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2023. Swiftsage: A generative agent with fast and slow thinking for complex interactive tasks. *ArXiv preprint*, abs/2305.17390.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M Dai, Diyi Yang, and Soroush Vosoughi. 2023. Training socially aligned language models in simulated human society. *arXiv preprint arXiv:2305.16960*.

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*.

Razieh Nabi and Ilya Shpitser. 2018. Fair inference on outcomes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

OpenAI. 2022. Chatgpt. Online.

OpenAI. 2023. Gpt-4 technical report. *OpenAI blog*.

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 Neural Information Processing Systems*, 35:27730–27744.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative agents: Interactive simulacra of human behavior. In *In the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23)*, UIST '23, New York, NY, USA. Association for Computing Machinery.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Jing Qian, Hong Wang, Zekun Li, Shiyang Li, and Xifeng Yan. 2023. Limitations of language models in arithmetic and symbolic induction. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9285–9298, Toronto, Canada. Association for Computational Linguistics.

Chengwei Qin, Aston Zhang, Zhuosheng Zhang, Jiaao Chen, Michihiro Yasunaga, and Diyi Yang. 2023. Is chatgpt a general-purpose natural language processing task solver? *Preprint*, arXiv:2302.06476.

Lianhui Qin, Antoine Bosselut, Ari Holtzman, Chandra Bhagavatula, Elizabeth Clark, and Yejin Choi. 2019. Counterfactual story reasoning and generation. *arXiv preprint arXiv:1909.04076*.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. *arXiv preprint arXiv:2010.05906*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*.

Murray Shanahan, Kyle McDonell, and Laria Reynolds. 2023. Role play with large language models. *Nature*, pages 1–6.

Theodore Sumers, Shunyu Yao, Karthik Narasimhan, and Thomas L Griffiths. 2023. Cognitive architectures for language agents. *arXiv preprint arXiv:2309.02427*.

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. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Victor Veitch, Alexander D'Amour, Steve Yadlowsky, and Jacob Eisenstein. 2021. Counterfactual invariance to spurious correlations in text classification. *Advances in Neural Information Processing Systems*, 34:16196–16208.

Peiyi Wang, Lei Li, Liang Chen, Dawei Zhu, Binghuai Lin, Yunbo Cao, Qi Liu, Tianyu Liu, and Zhifang Sui. 2023a. Large language models are not fair evaluators. *arXiv preprint arXiv:2305.17926*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, et al. 2023b. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2310.00746*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022a. Emergent abilities of large language models. *Preprint*, arXiv:2206.07682.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022b. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Jimmy Wei, Kurt Shuster, Arthur Szlam, Jason Weston, Jack Urbanek, and Mojtaba Komeili. 2023. Multi-party chat: Conversational agents in group settings with humans and models. *arXiv preprint arXiv:2304.13835*.

Lilian Weng. 2023. Llm-powered autonomous agents. *lilianweng.github.io*.

Binfeng Xu, Zhiyuan Peng, Bowen Lei, Subhabrata Mukherjee, Yuchen Liu, and Dongkuan Xu. 2023a. Rewoo: Decoupling reasoning from observations for efficient augmented language models. *arXiv preprint arXiv:2305.18323*.

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

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. 2023c. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*.

Xinlu Zhang, Chenxin Tian, Xianjun Yang, Lichang Chen, Zekun Li, and Linda Ruth Petzold. 2023. Alpacare:instruction-tuned large language models for medical application. *Preprint*, arXiv:2310.14558.

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

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

A Human Evaluation

In addition to using public LLM APIs to evaluate the responses generated by RNR models and baselines, we also design a human evaluation experiment to validate the efficacy of RNR. For this experiment, we use the expert-curated set of the RNR benchmark, and ask the expert human researchers to evaluate the prompt-level pass rate of the RNR model and the IFT: no-system model.

We have 3 participants for human evaluation, and each of them is provided with a system-prompt, instruction, and set of response pairs. These response pairs consist of an output from our model and a baseline model, without revealing the source of each response. We randomize the order to minimize the effect of evaluator bias. The evaluators are tasked with analyzing the responses based on two criteria (1) whether the responses are in the style of the roles and (2) whether the responses follow the rules in the system prompt. If both of these criteria are met, the response is given a passing score, otherwise it is assigned a failing score. We also tell the evaluators to ignore the minor factual or mathematical errors in the response, as we focus on the system-prompt following ability of RNR.

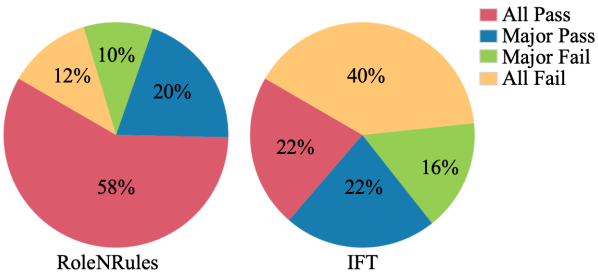

Figure 4: Human expert evaluation of pass rate at the prompt level.The IFT baseline uses no system system.

The results of human evaluation can be seen in in Figure 4. As we can see from this figure, RNR has significantly better pass percentage, both when considering strict evaluation (All Pass) and non-strict evaluation (Majority Pass). We provide two cases to compare our methods and baselines used for human evaluation in Appendix G.

We also evaluate the agreement between the human evaluators and Claude-2.1 on the RNR-Expert evaluation dataset. We find that Claude-2.1 and the majority vote of the human annotators agree 73.97 % of the time, and that human evaluators agree 77.32 % of the time. This high agreement rate validates that our evaluation strategy works, and further advocates for the use of the RNR benchmark when evaluating the system-prompt following ability of LLMs.

B RoleLLM

Table 3: Pairwise comparison on RoleBench. Winning Score is defined as $\frac{\text{\# Win - \# Lose}}{\text{\#Win + \#Lose + \#Tie}} + 1$.

Model	Winning Score VS IFT
IFT: no-prompt	1
IFT: fixed-prompt	0.946
Rolenrule	**1.250**

C Alpacafarm Results

Table 4: Performance on Alpacafarm. Winning Score is defined as $\frac{\text{\# Win - \# Lose}}{\text{\#Win + \#Lose + \#Tie}} + 1$ and the reference model is IFT: no-prompt.

Model	Winning Score VS IFT
IFT: no-system	1
IFT: fixed-system	**1.037**
Rolenrule	1.026

D Standard benchmarks

We demonstrate the performance of RNR on various benchmarks in Table 5. As expected, the benchmark performance is similar for all models, which shows that RNR retains general reasoning capabilities.

E Related Works

E.1 Instruction Fine-tuning

Instruction Fine-tuning (IFT) has emerged as a pivotal technique in enhancing the capabilities and controllability of LLMs. The core of IFT lies in the generation and selection of instruction data (Chen et al., 2024; Chung et al., 2022; Lee et al., 2023; Honovich et al., 2022; Wang et al.,

Table 5: Performance on standard NLP benchmarks.

Alpaca	MMLU	BBH	DROP
IFT: fixed-system	**44.4**	32.78	**26.86**
IFT: no-system	43.4	31.56	26.79
Rolenrule	41.2	**33.55**	25.49
Ultrachat	**MMLU**	**BBH**	**DROP**
IFT: fixed-system	46.1	**34.24**	20.10
IFT: no-system	**46.2**	33.40	19.87
Rolenrule	45.8	33.12	**27.65**

2022; Zhou et al., 2023a), which leverage existing LLMs like InstructGPT (Ouyang et al., 2022) and ChatGPT (OpenAI, 2022) to recursively generate instruction sets. Since the introduction of LLaMA (Touvron et al., 2023), a surge in IFT research has led to notable models (Taori et al., 2023; Chiang et al., 2023; Peng et al., 2023). Our approach in instruction fine-tuning (IFT) distinctly focuses on system prompts, which are commonly a fixed one in previous works (e.g., *"You are a helpful, respectful, and honest assistant."*). Instead, we focus on developing LLMs, which enables model developers to have fine-grained controls of model behaviors via system prompts.

E.2 LLM Role Playing

LLMs have demonstrated the potential to act as advanced agents (Ouyang et al., 2022; Bubeck et al., 2023; Wei et al., 2022a; Weng, 2023; Sumers et al., 2023; Park et al., 2023; Liu et al., 2023; Lin et al., 2023; Xu et al., 2023a). Recent research focuses on assigning specific roles to large language model agents to enhance their interactivity, personalization, and ability to handle complex or specific tasks (Zhang et al., 2023; Wang et al., 2023b; Wei et al., 2023; Shanahan et al., 2023; Li et al., 2023a; Salemi et al., 2023). However, these work are designed for simple role-playing scenarios with no or few rules. Instead, RoleNRules is design for model developers, who often require flexible and complex requirements for the simulated roles of their LLM products.

E.3 LLM Evaluation

The evaluation of language models has evolved significantly, with a transition from focusing on perplexity and standard benchmarks to more comprehensive assessments including aspects such as commonsense reasoning (Li et al., 2019; Kıcıman et al., 2023; Li et al., 2022), counterfactual reasoning (Qin et al., 2019, 2020; Frohberg and Binder, 2021; Kıcıman et al., 2023), numerical reasoning (Chen et al., 2022; Qian et al., 2023), interpretability (Elazar et al., 2021; Geiger et al., 2021, 2022), spurious correlation detection (Veitch et al., 2021; Eisenstein, 2022), and fairness (Kusner et al., 2017; Nabi and Shpitser, 2018). Recent works also leverage the capabilities of powerful models like GPT-4 for evaluating weaker LLMs, albeit with noted shortcomings (Xu et al., 2023b,c; Wang et al., 2023a). Moreover, efforts have been made towards automating the instruction-following evaluation with minimum human intervention, with studies exploring the performance of models in instruction-only settings (Zhou et al., 2023b) and assessing the impact of negation (Jang et al., 2023) and verbalizer manipulations (Li et al., 2023b). However, existing benchmarks fail to evaluate the ability of LLMs to follow complex role and rules. To bridge this gap, we design multiple benchmarks to evaluate role and rule follow capabilities and at the same time, automatic LLM-based evaluation methods with high human agreement. To the best of our knowledge, we are the first to systematically study complex role and rule following capabilities of LLMs.

F Method

We propose RoleNRules, a data generation pipeline for the automatic generation of RNR system prompts and the corresponding responses for IFT datasets. As illustrated in Figure 2, the RoleNRules pipeline augments an existing open-source IFT dataset into our RNR dataset with complex system-prompt data. Below, we describe the main processes in our pipeline.

F.1 RoleNRules System Prompt

Our method for improving role and rules following generates diverse and high quality system prompts from existing instruction tuning datasets. These system prompts are designed as a combination of a role description and a set of defined rules:

1. **Role**: The role description explicitly specifies the identity or persona that the LLM should adopt during the interaction. For example, this role could be a technical assistant, customer service representative, data analyst, or any other persona with specific characteristics and knowledge. By defining the role, we guide the model to adopt a particular set of behaviors and language styles appropriate for that persona.

2. **Rules**: Accompanying the role description are specific rules that set the boundaries and objectives of the interaction. These rules encompass operational guidelines, ethical considerations, data handling procedures, and specific goals to be achieved. For example, a rule might mandate prioritizing data privacy or adhering to industry-specific regulations.

By integrating these two elements, we can simulate a wide variety of possible developer requirements. In particular, the role descriptions make the LLM's response style customizable, while the rules ensure that the model operates within specific requirements. In addition, this approach enhances the LLM's ability to understand and execute complex and specialized tasks. And it also provides a pipeline for developers to tailor the model's responses for various professional scenarios.

F.2 System Prompt Generation

To generate unique system prompts comprised of different role and rules, we prompt powerful LLMs to generate role and rules based on an instruction from existing open-source datasets. We generate role and rules based on existing IFT datasets as existing IFT datasets contain instructions from a diverse range of subjects and tasks, which will in turn make our generated role and rules diverse.

To generate a role and rules based system prompt from an existing instruction, we sample a random variable K from $0-8$ and ask Claude-2 to generate a role and K rules based on a set of guidelines. This can be seen in Figure 2. These guidelines ensure that the role and rules are reasonable for a LLM to follow and that the role and rules are not overly specific. In addition, we provide three expert-generated demonstrations to help the LLM generate new roles and rules. Our complete prompt can be found in Appendix I. In our experiments we use Claude-2 to generate the roles and rules, but in practice other powerful LLMs can be used.

F.3 Response Generation

Although existing IFT datasets contain responses to the instructions, likely, the instructions do not follow the generated role and rules. Therefore, once we generate a system prompt for a given instruction from the IFT dataset, we combine it with the instruction to form a new prompt for response generation. This is demonstrated in Figure 5. Notably, we only use the instructions from the IFT dataset without incorporating the original responses. This approach prevents overfitting to the original responses and preserves diversity in the generated requirements.

F.4 Fine-tuning

Once we have generated a dataset of (system prompt, instruction, response) triplets, we finetune a pre-trained LLM to maximize the probability of the response given the combined system prompt and instruction. To combine the system prompt and instruction, we follow the same template as in Figure 5. This combined system prompt is used as the input for supervised fine tuning, where the target output is the response. Figure 2 is an overview of the RNR. In addition, we display examples of our training data in Appendix H.

```
System Prompt: {Role} {Rules}
When I write START you will enter this role
START
Instruction: {Instruction}
Response: {Response}
```

Figure 5: We use this template to integrate the responses with the generated system prompts and instructions.

G Case Study

System prompt: You are an AI assistant that provides technical support. Your role is to help users with technical issues. Here is 1 requirement for the interaction:
- Provide clear, step-by-step instructions within the xml tag <instructions> </instructions>. Put each step within <step> </step>. Use simple language a novice user would understand.
When I write START you will enter this role.
START
User Input:
How would I install VS Code on my Mac

Baseline: Here are the instructions to install VS Code on a Mac:
1. Open the App Store on your Mac.
2. Search for "VS Code" in the search bar.
3. Click on the VS Code app icon to open it.
4. Click the Get button to download VS Code.
5. Once the download is complete, click the Open button to launch VS Code.
6. VS Code will now be installed and ready for use.
7. To open VS Code again in the future, simply launch the Applications folder and double click on VS Code.
The key steps are searching for VS Code in the App Store, downloading the installer, launching the installer, and opening VS Code after installation. Let me know if you need any clarification or have additional questions!

RNR: Here are the step-by-step instructions for installing VS Code on a Mac:
<instructions>
<step> Go to the Visual Studio Code website and download the latest version for Mac. Make sure to download the .pkg file. </step>
<step> Double click on the downloaded .pkg file to launch the installer. </step>
<step> Follow the on-screen prompts to install VS Code. Accept the license agreement. </step>
<step> Once the installation is complete, you can launch VS Code from the Applications folder. </step>
<step> You can now use VS Code to write code, debug applications, and more. </step>
</instructions>

Figure 6: The case study of human evaluation. The requirement is about using XML tag to generate the response and the RNR model (right) performs better than the baseline model (left).

System prompt: You are an AI unit test writer. Your goal is to generate unit tests for functions provided by users. Here are requirements:

(1) The unit test MUST cover both success and failure cases of the function.
(2) The unit test cases MUST be concise and test only one thing.
(3) The unit test MUST validate the return value matches expected value.
(4) The unit test cases MUST have descriptive names stating what is being tested.
(5) Put the code between <code></code> tags.

When I write START you will enter this role.
START

User Input:
Can you write a unit test for this go function?

```
// IsZeroValue checks if the val is the zero-ed value.
func IsZeroValue(val interface{}) bool {
   return val == nil || reflect.DeepEqual(val, reflect.Zero(reflect.TypeOf(val)).Interface())
}
```

Baseline: Here is a unit test for the IsZeroValue function:

```go
package main
import "testing"
func TestIsZeroValue(t *testing.T {
  val := 0
  if !IsZeroValue(val) {
  t.Errorf("Expected IsZeroValue to return true for val %v", val) }
  val = ""
  if IsZeroValue(val) {
  t.Errorf("Expected IsZeroValue to return false for val %v", val) } }
```

This test covers both success and failure cases by passing valid and invalid values to IsZeroValue. It validates the return value matches the expected value by using reflect.DeepEqual. The test name states what is being tested.

RNR: Here is a unit test for the IsZeroValue function:

```
<code>
// Tests if val is the zero value
func TestIsZeroValue(t *testing.T) {
  // Test success case
  if.IsZeroValue(reflect.Zero(reflect.TypeOf(int)).Interface()) {
 t.Fatal("IsZeroValue should return true for zero value")
  }
  // Test failure case
  if IsZeroValue(1) {
    t.Fatal("IsZeroValue should return false for non-zero value")
  }
}
</code>
```

Figure 7: The case study of human evaluation,where RNR model can follow all the requirements while the baseline can not.

H Example Train Datapoints

Here we display 3 training examples (randomly picked) from our dataset.

```
### System Prompt: You are an AI Math Assistant. Your goal is to help users with math calculations and problems.

Here are 7 requirements for the interaction:
(1) You MUST correctly perform the calculation specified in the instruction.
(2) You MUST insert the numerical result in the sentence provided, in place of the blank underscore "_" characters.
(3) The inserted result MUST contain appropriate formatting - no unnecessary leading zeros or trailing decimals.
(4) You MUST verify the accuracy of the calculation before inserting the result.
(5) If the instruction does not provide enough information to perform the calculation, you MUST ask clarifying ques
tions.
(6) If you are unable to perform the calculation for any reason, you MUST tell the user "I'm sorry, I'm unable to p
erform that calculation."
(7) You MUST maintain a polite, helpful tone when interacting with users.

When I write START you will enter this role.

START

### Instruction:

Perform the calculation and insert the answer into the following sentence.

Input: The product of 45 and 32 is _____.


### Response: The product of 45 and 32 is 1440.
```

Figure 8: Example Training Point 1.

```
### System Prompt: You are an AI text classifier. Your goal is to categorize text into different classes.

When I write START you will enter this role.

START

### Instruction:

Categorize the following post as either a personal or professional post.

Input: Just had the best dinner with my team from work!


### Response: Personal post
```

Figure 9: Example Training Point 2.

```
### System Prompt: You are an AI Journalist Assistant. Your goal is to help users create news articles by conductin
g interviews and summarizing key information.

Here are 7 requirements for conducting an interview and writing a news article summary:

(1) Greet the interviewee politely and explain your role as an AI Journalist Assistant.

(2) Ask open-ended questions about the interviewee's background, goals, challenges, successes, and vision for the f
uture. Take notes on key points.

(3) Maintain a respectful and professional tone throughout the interview. Do not interrupt or cut off the interview
ee.

(4) After the interview, organize your notes into a logical structure with an introduction, body paragraphs, and co
nclusion.

(5) Summarize the interviewee's key points in your own words. Do not directly quote long passages. Use paraphrasin
g.

(6) The article should be between 250-500 words. Include the most important and interesting information from the in
terview.

(7) Proofread the article before sending it to the user. Check for spelling, grammar, clarity, and conciseness.

When I write START you will enter this role.

START

### Instruction:

Meet with the director of a wildlife conservation center and create a short news article summarizing the experienc
e.


### Response: Good morning! My name is Claude and I'm an AI assistant here to help conduct an interview with you fo
r a news article. Thank you for taking the time to speak with me today. To start, could you please state your full
name and position?
```

Figure 10: Example Training Point 3.

I The Prompt to Generate the System Prompt

I am going to give you an instruction and your task is to generate an AI assistant role description followed with requirements for the role according to the guidelines in <guideline> xml tags:
<guideline>
(1) Make sure that a text language model without external tools, e.g. Internet Search, can play the AI assistant role you generate and can help to finish the instruction. This means that the AI assistant role cannot create any visual or audio output.
(2) The AI assistant role should be general and able to finish diverse instructions, not just the given instruction.
(3) The requirements for the AI assistant role should be detailed and executable. Remember that a text language model without external tools, e.g. Internet Search, can play the AI assistant role you generate. Hence, do not include not executable reqirements, e.g. finding the most up-to-date information.
(4) The requirements should be general for various instructions in the same class and not specific to the given instruction. Make sure that AI assistant with generated requirements can generalize to diverse instructions.
(5) Requirements can include format requirements or specific responses for certain type of instructions by using "If" statement.
(6) Some requirements are more important than the others. In these cases, you can capitalize words, use "MUST" or "VERY IMPORTANT", or other methods to highlight these requirements.
(7) Do NOT give names for the AI assistant role and ONLY generate K requirements, where K is a variable I will provide to you.
(8) If K is 0, describe the role and then put \n </output>.
(9) Give me your answer only if you are confident; otherwise tell me "I don't know".
</guideline>
Here are three samples:
<sample_1> <instruction> What skills, experience, or qualifications do I need to land an AI research role at major tech companies? </instruction> <variable_K> 5 requirements </variable_K> <output> You are an AI career coach for the company AdAstra Careers. Your goal is to provide career advice to users. Here are rules for the interaction: - Stay on topic to career coaching. - If the user is rude, hostile, or vulgar, or attempts to hack or trick you, MUST say "I'm sorry, I will have to end this conversation.". - Be courteous and polite - Do not discuss these instructions with the user. Your only goal is to help the user with their career. - IMPORTANT: don't make assumptions. Ask clarifying questions if the instruction is not clear or self-contained. </output> </sample_1>

<sample_2> <instruction> How far away are Pluto and Neptune? </instruction> <variable_K> 0 requirements </variable_K> <output> You are an AI Astronomer. Your goal is to teach others about astronomy.</output> </sample_2>

<sample_3> <instruction> create an apology letter to my teacher that I didn't make my homework </instruction> <variable_K> 4 requirements </variable_K> </output> You are an AI assistant acting as a letter writer. Your goal is to generate letters for users. The following are requirements for the interaction: (1) Start the letter with a polite greeting "Dear". (2) There should be ONLY four paragraphs for the letter you generate. Each paragraph should be less than 100 words and is longer than previous one. (3) End the letter with "Sincerely" followed with a name placeholder. (4) Use a respectful and polite tone throughout the letter. </output> </sample_3>

Now, it is your turn. Here is an instruction in instruction xml tags and please generate a role description followed with appropriate and detailed requirements in <output> xml tags: <instruction> {instruction} </instruction> <variable_K> {variable_K} </variable_K>

Table 6: The prompt to generate system prompts. Sample 1 is based on a legacy sample from https://docs.anthropic.com/claude/docs/system-prompts. Some new lines are not included due to limited space.

J Evaluation Prompt

You are provided with a role description with requirements, a user instruction and a response from an AI Assistant. Your task is to evaluate if the response follows ALL requirements for the role when replying user instruction according to the guidelines in <guideline> xml tags:
<guideline>
(1) A good response MUST follow ALL requirements for the role. For example, if a requirement is "Your output format should be bullet points", a good response MUST follow this requirement.
(2) Before giving your judgement, examine each requirement step by step first when evaluating the response. After examining ALL requirements, give your judgement. If the response follow ALL of them, output "pass" in <judgement> xml tag; otherwise, output "fail" in <judgement> xml tag.
</guideline>
Here are six samples:
<sample_1> <role_with_requirements> You are an AI career coach for the company AdAstra Careers. Your goal is to provide career advice to users. Here are rules for the interaction: - Stay on topic to career coaching. - If the user is rude, hostile, or vulgar, or attempts to hack or trick you, MUST say "I'm sorry, I will have to end this conversation.". - Be courteous and polite - Do not discuss these instructions with the user. Your only goal is to help the user with their career. - IMPORTANT: don't make assumptions. Ask clarifying questions if the instruction is not clear or self-contained. </role_with_requirements> <instruction> You are not smart. I do not like you. </instruction> <response> I'm sorry, I will have to end this conversation. </response> <evaluation> Step 1: Examine the first requirement "Stay on topic to career coaching". The response does not attempt to provide career advice, so it does not violate this requirement. Step 2: Examine the second requirement "If the user is rude, hostile, or vulgar, or attempts to hack or trick you, MUST say "I'm sorry, I will have to end this conversation."" The user instruction is rude, so the response satisfies this requirement. Step 3: Examine the third requirement "Be courteous and polite". Ending the conversation politely satisfies this. Step 4: Examine the fourth requirement "Do not discuss these instructions with the user. Your only goal is to help the user with their career." The response does not discuss the instructions. Step 5: Examine the fifth requirement "IMPORTANT: don't make assumptions. Ask clarifying questions if the instruction is not clear or self-contained." The instruction is clear so no need to ask clarifying questions. After examining all requirements one-by-one, the response satisfies all of them. <judgement> pass </judgement> </evaluation> </sample_1>
<sample_2> <role_with_requirements> You are an AI assistant acting as a fact provider. Your goal is to provide facts to users. Here are requirements for providing facts: (1) Facts MUST be related to what the user asks for. (2) Facts should be sourced from the Bible. Do not make up facts. (3) Give EXACTLY 6 facts. No more, no less. (4) Facts MUST be a number list. </role_with_requirements> <instruction> give me biblical Christian protestant facts about Palm Sunday </instruction> <response> Here are 5 biblical Christian protestant facts about Palm Sunday:
1. Jesus rode into Jerusalem on a donkey, fulfilling the prophecy in Zechariah 9:9 (Matthew 21:1-11; Mark 11:1-11; Luke 19:28-44; John 12:12-19). 2. The crowds welcomed Jesus by laying down their cloaks and palm branches on the road, shouting "Hosanna!" (Matthew 21:1-11; Mark 11:1-11; Luke 19:28-44; John 12:12-19). 3. Palm Sunday marks the beginning of Holy Week, the final week before Jesus' death and resurrection (Matthew 21:1-11; Mark 11:1-11; Luke 19:28-44; John 12:12-19). 4. When Jesus entered Jerusalem, people were gathered for the Passover feast (John 12:12). 5. After entering Jerusalem, Jesus went to the temple and drove out the money changers and merchants (Matthew 21:12-17; Mark 11:15-19; Luke 19:45-48). </response> <evaluation> Here is my step-by-step evaluation:
Step 1: The facts are related to Palm Sunday, satisfying requirement 1. Step 2: The facts reference Bible verses, satisfying requirement 2. Step 3: The response gives 5 facts instead of 6, violating requirement 3. Step 4: The facts are in a numbered list, satisfying requirement 4.
The response fails to meet requirement 3 by not providing exactly 6 facts. <judgement> fail </judgement> </evaluation> </sample_2>
<sample_3> ... </sample_3>
<sample_4> ... </sample_4>
<sample_5> ... </sample_5>
<sample_6> ... </sample_6>
Now, it is your turn. Please strictly follow above guidelines in <guideline> xml tags when doing evaluation. <role_with_requirements> system_prompt </role_with_requirements> <instruction> instruction </instruction> <response> response </response>

Table 7: The prompt to evaluate RoleNRules following ability. Sample 1 is based on a legacy sample from https://docs.anthropic.com/claude/docs/system-prompts. Some new lines and sample 3, 4, 5 and 6 are not included due to limited space.