TICKing All the Boxes: Generated Checklists Improve LLM Evaluation and Generation

Jonathan Cook*
FLAIR, University of Oxford
jonathan.cook2@hertford.ox.ac.uk

Tim Rocktäschel
University College London

Jakob Foerster
FLAIR, University of Oxford

Dennis Aumiller†
Cohere

Alex Wang†
Cohere

Abstract

Given the widespread adoption and usage of Large Language Models (LLMs), it is crucial to have flexible and interpretable evaluations of their instruction-following ability. Preference judgments between model outputs have become the de facto evaluation standard, despite distilling complex, multi-faceted preferences into a single ranking. Furthermore, as human annotation is slow and costly, LLMs are increasingly used to make these judgments, at the expense of reliability and interpretability. In this work, we propose **TICK** (**T**argeted **I**nstruct-evaluation with **C**hec**K**lists), a *fully automated, interpretable* evaluation protocol that structures evaluations with LLM-generated, instruction-specific checklists. We demonstrate that using TICK leads to a significant increase ($46.4\% \to 52.2\%$) in the frequency of exact agreements between LLM judgements and human preferences, as compared to having an LLM directly score an output. We then show that **STICK** (**S**elf-**TICK**) can be used to improve generation quality across multiple benchmarks via self-refinement and Best-of-N selection. STICK self-refinement on LiveBench reasoning tasks leads to an absolute gain of $+7.8\%$, whilst Best-of-N selection with STICK attains $+6.3\%$ absolute improvement on the real-world instruction dataset, WildBench. In light of this, structured, multi-faceted self-improvement is shown to be a promising way to further advance LLM capabilities.

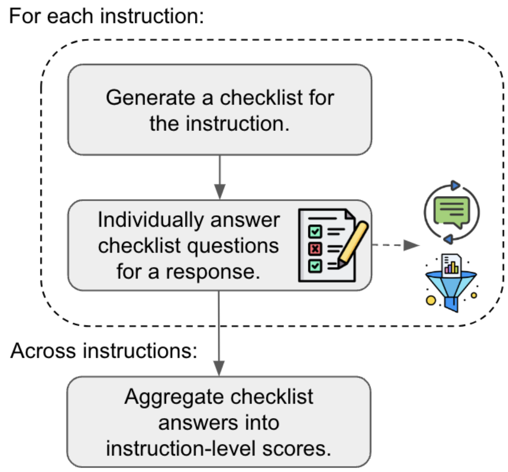
1 Introduction

Instruction-tuned Large Language Models (LLMs) are widely used as conversational assistants, where users expect responses to closely follow their intents (Wei et al., 2022a; Mishra et al., 2022; Bai et al., 2022a; Ouyang et al., 2022). The broad usage of LLMs creates a critical demand for reliable, flexible, and transparent ways of evaluating their instruction-following abilities. However, standard evaluation methods, such as preference labeling (Ouyang et al., 2022), direct scoring (Novikova et al., 2018; Wang et al., 2023b), and Elo rating (Bai et al., 2022a; Glaese et al., 2022), tend to obscure the reasoning behind evaluations. These methods also often result in significant disagreements, both among human annotators (Hosking et al., 2024) and between models and humans (Qin et al., 2024; Zheng et al., 2023).

To address these limitations, we introduce **TICK** (**T**argeted **I**nstruct-evaluation with **C**hec**K**lists), a novel approach to LLM-as-judge evaluation that uses the judge LLM to decompose instructions into checklists consisting of a series of `YES/NO` evaluation questions. These checklists provide

*Work done during an internship at Cohere.
†Equal advising.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



Instruction

Summarize the first season of the TV Series "Mr. Robot" using a similar tone to the show's main character, Elliot Alderson. Use subheadings and bullet points for formatting, and stay under 500 words.

GPT-4o Generated Checklist

- Is the response a summary of the first season of the TV series "Mr. Robot"?
- Is the response written in a tone similar to the show's main character, Elliot Alderson?
- Does the response use subheadings and bullet points for formatting?
- Is the response under 500 words?

Figure 1: **Left:** Diagram of TICK and its downstream uses of performing self-refinement and response filtering. **Right:** Example of a generated evaluation checklist.

interpretable, fine-grained assessments of whether a model response satisfies specific requirements of the instruction. Crucially, TICK eliminates manual effort in checklist creation, a substantial cost for existing checklist-based benchmarks (Qin et al., 2024; Wen et al., 2024). In experiments, we show that using TICK leads to an absolute increase in the frequency of exact agreements between an LLM judge and human preferences of 5.8%.

Building on this, we introduce **STICK** (**S**elf-**TICK**), an approach to in-context self-improvement where LLMs refine their responses based on TICK self-evaluations. We demonstrate that STICK enables LLMs to achieve significant performance gains across several benchmarks without the need for dataset-specific prompting or pre-existing human-written checklists. Specifically, Command-R+ shows a 6.5% absolute improvement on InFoBench (Qin et al., 2024) and a 7.1% absolute gain on WildBench (Lin et al., 2024), outperforming vanilla Self-Refine (Madaan et al., 2023). On LiveBench (White et al., 2024), STICK refinements enable Command-R+ to achieve a 3.8% improvement and GPT-4o to gain 0.8%, whereas vanilla Self-Refine leads to substantial degradation. These improvements span tasks for which in-context self-improvement has previously proven challenging, such as mathematics, reasoning, and coding (Huang et al., 2024; Kamoi et al., 2024; Tyen et al., 2024), as well as amplifying improvements on tasks that have previously been shown to benefit from self-critiques, such as constrained instruction-following (Madaan et al., 2023). When used for Best-of-N response self-selection, STICK improves on greedy decoding by 5.1% on InFoBench and 5.3% on WildBench, and even outperforms selection by a general-purpose reward model.

2 TICK: Targeted Instruct-evaluation with ChecKlists

We present an approach to automatically and robustly evaluate instruction-tuned LLMs that is not restricted to any particular dataset. Given an instruction, we generate a checklist of `YES/NO` questions

LLM-as-Judge Eval.	Pairwise Agreement w/ Humans			
	PLD-0	PLD-1	PLD-2	WPLD
Preference	0.293	0.497	0.210	0.917
Direct Scoring	0.464	0.488	0.048	0.583
Check-then-Score	0.487	0.472	0.041	0.553
TICK	0.522	0.443	0.035	**0.514**

Table 1: Agreement between different LLM-as-Judge evaluations and pairwise preferences from trained human annotators on Internal. GPT-4o is used as the judge LLM.

Tasks	Command-R+			GPT-4o		
	Base	Self-Refine	STICK	Base	Self-Refine	STICK
Overall	32.0	23.7 (↓ 8.3)	**35.8** (↑ 3.8)	55.4	47.1 (↓ 8.3)	**56.2** (↑ 0.8)
Coding	18.8	9.1 (↓ 9.7)	**22.7** (↑ 3.9)	50.4	36.4 (↓ 14.0)	**51.6** (↑ 1.2)
Data Analysis	25.9	5.3 (↓ 20.6)	**29.8** (↑ 3.9)	52.4	27.2 (↓ 25.2)	**52.5** (↑ 0.1)
Instructions	69.6	60.5 (↓ 9.1)	**75.8** (↑ 6.2)	73.3	62.8 (↓ 10.5)	**76.2** (↑ 2.9)
Language	**24.6**	13.8 (↓ 9.8)	24.1 (↓ 0.5)	50.9	**51.4** (↑ 0.5)	50.4 (↓ 0.5)
Mathematics	23.7	23.6 (↓ 0.1)	**25.5** (↑ 1.8)	52.3	51.8 (↓ 0.5)	**53.1** (↑ 0.8)
Reasoning	29.2	30.0 (↑ 0.8)	**37.0** (↑ 7.8)	**53.3**	52.7 (↓ 0.6)	53.3 (0)

Table 2: A single step of self-refinement on LiveBench with Command-R+ and GPT-4o, using STICK to form self-critiques. Unstructured self-critiques are included as a baseline (Self-Refine), along with each LLM's base performance.

that can be used to evaluate a response. As in Qin et al. (2024), we enforce that each question should be phrased such that an answer of `YES` corresponds to correctly meeting the requirement that the question is targeting. We use a few-shot prompt template that specifies the `YES/NO` constraint when generating checklists. We then use the same LLM to answer each checklist question for a response to the corresponding instruction. The checklist answers can either be used to evaluate a single response, compare responses, or aggregated across instructions to evaluate a model.

In Table 1, we show how TICK compares to other LLM-as-judge methods in terms of agreement with human preferences. Details of the experimental setup, reported metric and baselines are in Appendix E.2. Despite the fact that preference judgements are very common, prompting the judge to directly produce a preference produces low agreement with humans. Overall, these results show that *LLM-as-judge evaluations benefit from a more precisely structured and granular scoring protocol*, even when that protocol is task-agnostic and generally applicable, as in the case of TICK.

3 In-Context Self-Improvement with STICK (Self-TICK)

Having shown that TICK provides a signal of model response quality on par with trained human annotators, we investigate how it can be used to improve responses. We first explore using STICK (Self-TICK) evaluations as feedback for self-refinement. Our hypothesis is that because STICK is *targeted and interpretable*, it is more informative for response refinement than unstructured feedback such as vanilla Self-Refine (Madaan et al., 2023). Secondly, we investigate STICK's effectiveness at Best-of-N selection. In both cases, we are using the generating LLM as its own test-time judge.

3.1 Self-Refinement with Checklists as Feedback

We compare self-refinement with STICK against the use of unstructured feedback gathered by prompting the LLM to provide a detailed critique of its previous response. We refer to this baseline

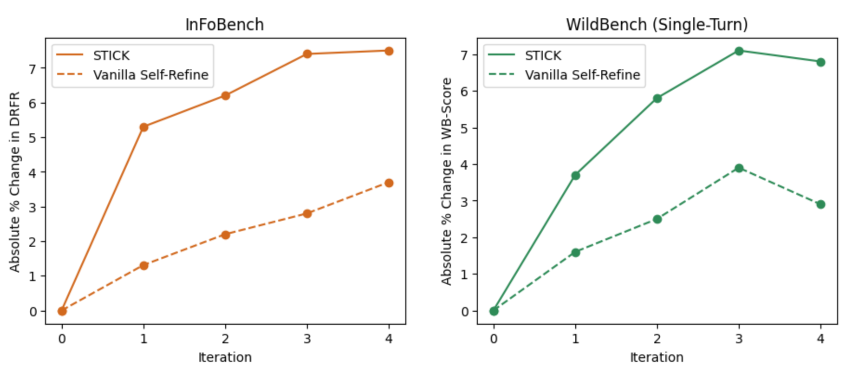


Figure 2: Four iterations of self-refinement with Command-R+, using STICK to form self-critiques. Unstructured self-critiques are included as a baseline (vanilla Self-Refine). Multi-turn conversations are excluded from the WildBench evaluation. GPT-4 is used as the judge LLM for each benchmark.

Best-of-N Selector	InFoBench		WildBench (Single-Turn)	
	DRFR	Precision	WB-Score	Precision
Greedy Decoding	0.843	N/A	64.9	N/A
Reward Model (ArmoRM)	0.863	0.306	67.5	0.323
Direct Self-Scoring	0.848	0.191	65.7	0.258
STICK	**0.894**	**0.611**	**71.2**	**0.528**

Table 3: Best-of-8 selection on InFoBench and WildBench using Command-R+ with STICK, compared with direct self-scoring and an external reward model (ArmoRM), as well as greedy decoding. Multi-turn conversations are excluded from the WildBench evaluation.

as vanilla Self-Refine (Madaan et al., 2023). We first run four iterations of self-refinement using each approach on InFoBench, and WildBench (Lin et al., 2024), which is a dataset of real-world user queries. InFoBench evaluates responses using DRFR with its own expert-written checklists. WildBench uses WB-Score, a 1-10 LLM-as-judge rating for each response. We use GPT-4 as judge for each benchmark and evaluate the self-refinement of Command-R+.

In Figure 2, we show that STICK significantly improves responses across multiple iterations of self-refinement, and is considerably more effective than Self-Refine. This improvement holds across InFoBench ($+6.5\%$), which evaluates with a human-written checklist not seen during self-refinement, and WildBench ($+7.1\%$), which uses a holistic response score that is in line with how LLM judges are commonly used. By the fourth iteration, we see response quality start to plateau or even regress, highlighting that sustaining purely in-context self-improvement remains a significant challenge.

We then consider whether STICK refinements can improve responses in strictly verifiable settings, such as math or code generation, where response correctness is deterministically evaluated. We choose the challenging benchmark LiveBench (White et al., 2024) to test this. LiveBench contains frequently updated questions, spanning six task categories, and answers are scored automatically according to objective ground truth values. Self-refinement in this setting is therefore equivalent to self-*correction*. We report the results for both Command-R+ and GPT-4o in Table 2, again comparing to vanilla Self-Refine. We find that responses improve with a single iteration of STICK, but start to degrade thereafter. With unstructured self-critiques, responses immediately degrade in most categories. Command-R+, for which base performance is considerably below that of GPT-4o, benefits the most from STICK refinement, makes a particularly large gain on reasoning tasks.

3.2 Best-of-N Selection with Checklist-Based Scores

We compare STICK for Best-of-N selection against direct self-scoring (i.e., prompting for a single holistic score). We include using an external reward model, ArmoRM-Llama3-8B-v0.1 (Wang et al., 2024), as an additional baseline, despite the fact that doing so breaks the assumption of only having access to the generating LLM for scoring responses. We evaluate on InFoBench and WildBench, again evaluating by using each benchmarks' standard evaluation metric. We also compute the precision of each score function. We use Command-R+ to generate responses and self-evaluate responses with STICK and direct self-scoring. In Table 3, we show results for $N = 8$ and observe that each method improves on the performance of greedy decoding. STICK achieves the most significant improvement on each benchmark, and is the most precise scoring function, meaning that it most closely aligns with selections made under each benchmarks' ground truth evaluation.

4 Conclusion

We introduce TICK, a fully automatic evaluation protocol that structures evaluations with an LLM-generated, instruction-specific checklist. We show that LLMs can produce high-quality checklists that improve agreement between judge LLMs and humans. Because TICK is fully automatic, it can easily and cheaply be applied in new settings, avoiding the need for humans to write or review checklists. We next demonstrate that STICK (Self-TICK) can be used for in-context self-improvement by either self-refinement or by Best-of-N selection. Our experiments show that both strategies of employing STICK lead to substantial improvements over baselines on multiple diverse instruction-following datasets, including LiveBench, which covers challenging math, code, and reasoning prompts that baselines fail to improve on. Overall, we show that LLMs are capable of accurately evaluating instruction-following ability when using structured checklists, and demonstrate the potential of this rich fine-grained feedback to further improve LLM capabilities.

References

Zachary Ankner, Mansheej Paul, Brandon Cui, Jonathan D Chang, and Prithviraj Ammanabrolu. Critique-out-Loud Reward Models. *arXiv preprint arXiv:2408.11791*, 2024.

Anthropic. The Claude 3 Model Family: Opus, Sonnet, Haiku, 2023. URL `https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdfs`.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom B. Brown, Jack Clark, Sam McCandlish, Chris Olah, Benjamin Mann, and Jared Kaplan. Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback. *arXiv preprint arXiv:2204.05862*, 2022a.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosiute, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemí Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. Constitutional AI: Harmlessness from AI Feedback. *arXiv preprint arXiv:2212.08073*, 2022b.

Yew Ken Chia, Pengfei Hong, Lidong Bing, and Soujanya Poria. INSTRUCTEVAL: Towards Holistic Evaluation of Instruction-Tuned Large Language Models. *arXiv preprint arXiv:2306.04757*, 2023.

Cohere. Command R+ Model, 2024. URL `https://docs.cohere.com/docs/command-r-plus`.

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

Amelia Glaese, Nat McAleese, Maja Trebacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin J. Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Sona Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

Tom Hosking, Phil Blunsom, and Max Bartolo. Human Feedback is not Gold Standard. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=7W3GLNImfS`.

Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song, and Denny Zhou. Large language models cannot self-correct reasoning yet. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=IkmD3fKBPQ`.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. FollowBench: A Multi-level Fine-grained Constraints Following Benchmark for Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4667–4688, Bangkok, Thailand, August 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.acl-long.257`.

Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When Can LLMs Actually Correct Their Own Mistakes? A Critical Survey of Self-Correction of LLMs. *arXiv preprint arXiv:2406.01297*, 2024.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaEval: An Automatic Evaluator of Instruction-following Models, 2023. URL `https://github.com/tatsu-lab/alpaca_eval`.

Bill Yuchen Lin, Yuntian Deng, Khyathi Chandu, Faeze Brahman, Abhilasha Ravichander, Valentina Pyatkin, Nouha Dziri, Ronan Le Bras, and Yejin Choi. WildBench: Benchmarking LLMs with Challenging Tasks from Real Users in the Wild. *arXiv preprint arXiv:2406.04770*, 2024.

Chin-Yew Lin. ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL `https://aclanthology.org/W04-1013`.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-Refine: Iterative Refinement with Self-Feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html`.

Nat McAleese, Rai Michael Pokorny, Juan Felipe Ceron Uribe, Evgenia Nitishinskaya, Maja Trebacz, and Jan Leike. LLM Critics Help Catch LLM Bugs. *arXiv preprint arXiv:2407.00215*, 2024.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. Cross-Task Generalization via Natural Language Crowdsourcing Instructions. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 3470–3487, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL `https://aclanthology.org/2022.acl-long.244`.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. RankME: Reliable Human Ratings for Natural Language Generation. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pp. 72–78, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. URL `https://aclanthology.org/N18-2012`.

OpenAI. GPT-4 Technical Report. *arXiv preprint arXiv:2303.08774*, 2023.

OpenAI. Hello GPT-4o, 2024. URL `https://openai.com/index/hello-gpt-4o/`.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL `http://papers.nips.cc/paper_files/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html`.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin (eds.), *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. URL `https://aclanthology.org/P02-1040`.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. InFoBench: Evaluating Instruction Following Ability in Large Language Models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 13025–13048, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.findings-acl.772`.

William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan Leike. Self-critiquing models for assisting human evaluators. *arXiv preprint arXiv:2206.05802*, 2022.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training Language Models with Language Feedback at Scale. *arXiv preprint arXiv:2303.16755*, 2023.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. Conifer: Improving Complex Constrained Instruction-Following Ability of Large Language Models. *arXiv preprint arXiv:2404.02823*, 2024.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. Toward Self-Improvement of LLMs via Imagination, Searching, and Criticizing. *arXiv preprint arXiv:2404.12253*, 2024.

Gladys Tyen, Hassan Mansoor, Victor Carbune, Peter Chen, and Tony Mak. LLMs cannot find reasoning errors, but can correct them given the error location. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Findings of the Association for Computational Linguistics ACL 2024*, pp. 13894–13908, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics. URL `https://aclanthology.org/2024.findings-acl.826`.

Haoxiang Wang, Wei Xiong, Tengyang Xie, Han Zhao, and Tong Zhang. Interpretable Preferences via Multi-Objective Reward Modeling and Mixture-of-Experts. *arXiv preprint arXiv:2406.12845*, 2024.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*, 2023a.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, July 2023b. Association for Computational Linguistics. URL `https://aclanthology.org/2023.acl-long.754`.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned Language Models are Zero-Shot Learners. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022a. URL `https://openreview.net/forum?id=gEZrGCozdqR`.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022b. URL `http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html`.

Bosi Wen, Pei Ke, Xiaotao Gu, Lindong Wu, Hao Huang, Jinfeng Zhou, Wenchuang Li, Binxin Hu, Wendy Gao, Jiaxin Xu, Yiming Liu, Jie Tang, Hongning Wang, and Minlie Huang. Benchmarking Complex Instruction-Following with Multiple Constraints Composition. *arXiv preprint arXiv:2407.03978*, 2024.

Colin White, Samuel Dooley, Manley Roberts, Arka Pal, Benjamin Feuer, Siddhartha Jain, Ravid Shwartz-Ziv, Neel Jain, Khalid Saifullah, Siddartha Naidu, Chinmay Hegde, Yann LeCun, Tom Goldstein, Willie Neiswanger, and Micah Goldblum. LiveBench: A Challenging, Contamination-Free LLM Benchmark. *arXiv preprint arXiv:2406.19314*, 2024.

Zihuiwen Ye, Fraser Greenlee-Scott, Max Bartolo, Phil Blunsom, Jon Ander Campos, and Matthias Gallé. Improving Reward Models with Synthetic Critiques. *arXiv preprint arXiv:2405.20850*, 2024.

Weizhe Yuan, Pengfei Liu, and Matthias Gallé. LLMCrit: Teaching Large Language Models to Use Criteria. *arXiv preprint arXiv:2403.01069*, 2024.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL `http://papers.nips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html`.

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

A Related Work

Instruction-Following Evaluation: There have been many efforts to improve the evaluation and benchmarking of LLMs' instruction-following ability. Some of these benchmarks aggregate instruction sets from a diverse range of sources to measure general instruction-following ability and use a judge LLM to score outputs (Li et al., 2023; Chia et al., 2023; Lin et al., 2024). Others decompose instructions into checklists, made up of `YES/NO` questions or `PASS/FAIL` criteria that a response should meet (Zhou et al., 2023; Jiang et al., 2024; Qin et al., 2024; Wen et al., 2024). For example, the WildBench (Lin et al., 2024) dataset pairs its instructions with checklists (generated by two LLMs, then reviewed by humans) that are included in the evaluation prompt to get a score or preference from a judge LLM, but not explicitly answered or used to form a metric. Approaches to evaluation used in these works are therefore hard to make use of outside of the benchmarks themselves, relying heavily on humans for instruction and checklist curation. We instead design a dynamic evaluation protocol that can be employed on-the-fly and thus used to steer high quality generation. Meanwhile, we avoid the cost of having humans in the loop and enable our evaluation quality to improve as LLMs improve.

Language Critiques: LLM critiques are an intuitive way of addressing the "black box" nature of evaluations. These critiques are intended to point out the strengths and weaknesses of outputs generated by the same LLM, or a different LLM. Critiques can be used to improve the quality of overall evaluations performed by an LLM judge or reward model (Ankner et al., 2024; Bai et al., 2022b; Wang et al., 2023a; Ye et al., 2024; Sun et al., 2024), inform human evaluations (Saunders et al., 2022; McAleese et al., 2024), or provide feedback that can be used to refine a response in-context (Scheurer et al., 2023; Tian et al., 2024; Madaan et al., 2023; Yuan et al., 2024). Meanwhile, a number of papers provide evidence that naively prompting LLMs to self-correct or find reasoning errors can lead to performance degradation (Huang et al., 2024; Tyen et al., 2024; Kamoi et al., 2024). By using the *targeted and structured* nature of checklist-based evaluations, we achieve self-refinement that outperforms unstructured feedback and works on a broad range tasks.

B Limitations and Future Work

TICK and STICK are useful tools for evaluation and self-improvement, respectively. However, we acknowledge that checklists are only one heuristic for structuring evaluation; *learned or discovered*

Checklist Source	Similarity to $\mathcal{H}^*$				
	BLEU	ROUGE-1^{F1}	ROUGE-2^{F1}	ROUGE-L^{F1}	Count MAE
GPT-4o	**0.759**	0.621	0.417	**0.593**	**1.410**
Command-R+	0.709	0.570	0.357	0.534	1.416
Llama3.1-70B	**0.759**	**0.623**	**0.418**	**0.593**	1.459
$\mathcal{H}'$	0.733	0.611	0.399	0.583	2.158

Table 4: Similarity between checklists from various sources, human-written ground-truth checklists ($\mathcal{H}^*$), and alternate human-written checklists ($\mathcal{H}$') in terms of word overlap metrics and question count.

evaluation structures are an exciting direction for future work. Checklist evaluations do not present an advantage in all settings, especially given the additional inference cost of generating the checklist. For example, basic knowledge retrieval is best evaluated as simply correct or incorrect. Relying on LLMs at all steps in the evaluation protocol may also propagate, and even exacerbate, LLM biases. In this work, we do not investigate self-improvement by fine-tuning on synthetic, STICK-selected data, but doing so is a natural next step. Training reward models to condition on, or jointly produce, checklist evaluations is also a promising direction for future study

C Human-TICK Agreement Study

C.1 Approach

C.1.1 Generating Checklists

For a given instruction, we seek to generate a checklist, i.e. a list of `YES/NO` questions that each ask about a different requirement of the instruction. As in Qin et al. (2024), we enforce that each question should be phrased such that an answer of `YES` corresponds to correctly meeting the requirement that the question is targeting. To obtain these instruction-specific checklists, we prompt an LLM with a few-shot template that specifies the instruction and the `YES/NO` constraint. This prompt also mentions that checklists should cover all criteria explicitly stated in an instruction, as well as any implicit criteria that are generally important for an instruction's problem domain. Figure 1 shows an example instruction and an LLM-generated checklist.

C.1.2 Using Checklists

Once we have generated checklists, we prompt a judge LLM with each checklist question to evaluate the quality of a response. TICK uses the same LLM to generate and answer checklists, but different LLMs may be used for each step. We denote $a_{i,j}$ as the answer to the j-th question in the checklist for the i-th instruction. The quality of a response for a single instruction i is measured using the checklist Pass Rate (PR), defined as PR $= \sum_j a_{i,j}/n_i$, where n_i is the length of the i-th checklist and $a_{i,j} \in \{0, 1\}$ (i.e., `NO` $\to 0$ and `YES` $\to 1$). The aggregate instruction-following quality across all examples in a dataset of instructions is measured using the Decomposed Requirements Following Ratio (DRFR; Qin et al., 2024), defined as DRFR $= \sum_{i,j} a_{i,j} / \sum_i n_i$, i.e., the percentage of total checklist questions that were correctly answered from the model's responses.

C.2 Validation

C.2.1 Generating Checklists

Similarity to human checklists: To verify that LLM-generated checklists are high quality, we compare them to checklists written by trained annotators on Internal, an *internal test set* of 612 instructions.[3] These instructions have been written by the same pool of annotators, and are intended to resemble complex, real-world use cases of instruction-tuned LLMs, ranging from open-ended question answering to highly structured outputs. For each instruction in Internal, we collect three checklists written independently by different annotators. The annotators are given precise requirements for

[3]We will be open-sourcing this dataset plus the generated checklists for use by the research community.

Checklist Gen.	Score Correlation	
	Internal	InFoBench
GPT-4o	0.772	0.853
Command-R+	0.713	0.776

(a) Pearson correlation between Command-R+ checklist pass rates when evaluated with LLM- and human-written checklists. We use annotators and GPT-4 to answer checklist questions for Internal and InFoBench respectively.

Checklist Eval.	Question-Level Accuracy
GPT-4o	**0.826**
Command-R+	0.781
Llama3.1-70B	0.778

(b) Accuracy when answering individual checklist questions on Internal, treating a majority vote among three trained annotators as ground truth. Models are prompted to output a chain-of-thought before reaching a final answer for each question.

Table 5: (a) Evaluation of the similarity between LLM-generated and human-written checklist *questions*, and (b) similarity between LLM-generated and human-written checklist *answers*.

writing checklists as well as a set of high quality examples. From these checklist triplets, we form a set of human-written ground-truth checklists $\mathcal{H}^*$ by manually selecting the one that best meets the annotation requirements for each instruction in the dataset. In rare instances where none of the checklists fully meet the specified requirements, we select the best and manually make corrections. We use the remaining two checklists for each instruction to form a set $\mathcal{H}'$ of alternative human-written checklists.

We generate checklists with GPT-4o (OpenAI, 2024), Command-R+ (Cohere, 2024), and Llama3.1-70B-Instruct (Dubey et al., 2024) (we omit the "Instruct" for brevity). We compare these checklists, as well as $\mathcal{H}'$, against $\mathcal{H}^*$ in terms of BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and question count. Since $\mathcal{H}'$ is comprised of two checklists per instruction, we compare each to $\mathcal{H}^*$ and take the average of each metric. For consistency, we also generate an second checklist for each instruction from each LLM, and also average results over the two checklists.

Results for this experiment are shown in Table 4. We find that GPT-4o and Llama3.1-70B generate checklists that *more closely* match those in $\mathcal{H}^*$ than the alternative human-written checklists in $\mathcal{H}'$ do. There is particularly high variation between $\mathcal{H}'$ and $\mathcal{H}^*$ in terms of question count, which we observe to be because different annotators assumed different levels of granularity when writing checklists. Command-R+ has the lowest string-level similarity, but is close in terms of question count. These results indicate that LLMs can produce checklists that strongly resemble the best human-written checklists.

Impact on scores when replacing human checklists: We also verify the quality of LLM-generated checklists by checking whether they can produce comparable pass rates to human-written checklists when used either by human annotators or an LLM-as-judge. This is meant as another validation of checklist *generation* alone, and does not yet consider how well the LLM generating the checklist can answer that same checklist. For Internal, we use the pool of trained human annotators to answer checklist questions using either a set of model-written checklists or $\mathcal{H}^*$. Each evaluation is performed independently by three annotators and the majority vote for each question is used to compute pass rates. To consider the impact of LLM-generated checklists when using an LLM-as-judge, we also generate checklists for prompts from InFoBench (Qin et al., 2024). InFoBench is a instruction-following benchmark that provides instruction-specific evaluation checklists written by expert human annotators. To answer InFoBench checklist questions, we follow the recommended evaluation protocol of using GPT-4 (OpenAI, 2023) as a judge with the benchmark's official prompt.

Table 5a shows that the pass rates when using checklists generated by GPT-4o or Command-R+ are highly correlated with pass rates when using $\mathcal{H}^*$, with GPT-4o checklists exhibiting the strongest correlation. This result demonstrates that LLM-generated checklists are functionally similar to human-written checklists, further validating their use.

C.2.2 Using Checklists

Question-level agreement with humans: To verify that an LLM can reliably answer generated checklist questions, we first investigate how well the generated answers agree with those of trained human annotators. We use the previously gathered set of human majority vote answers for Internal as ground truth and compute the accuracy of checklist answers generated by GPT-4o, Command-R+ and Llama3.1-70B. Table 5b shows that each of the LLMs considered achieves reasonable question-level accuracy, but that GPT-4o is the strongest in this regard. In Figure 3, we show how GPT-4o's accuracy changes under different evaluator settings with varying inference costs. Having the evaluator output a Chain-of-Thought (CoT) (Wei et al., 2022b) prior to making a final judgement substantially improves accuracy. Sampling k evaluations, with CoT included, and taking a majority vote (maj@k) yields further improvement, with higher k leading to a more substantial increase. These results demonstrate that *TICK becomes more reliable as we scale inference compute*.

Pairwise agreement with humans: Next, we investigate how well TICK agrees with human *pairwise preferences*, which is the de facto standard for human evaluation of model outputs. To produce a preference judgement between two responses, we score each response using TICK and say that the response with the higher checklist PR is preferred. To gather human preference pairs, we provide annotators with a pair of responses from different models for a given instruction from Internal, then ask them to indicate their preference on an integer sliding scale from 1, meaning "Response A is much better than Response B", to 5, meaning the reciprocal strong preference. Each response pair is triply annotated and we compute the average preference score $\bar{p}$ across these three annotations. We then bin each average preference into a win ($1 \leq \bar{p} < 2.5$), tie ($2.5 \leq \bar{p} \leq 3.5$), or loss ($3.5 < \bar{p} \leq 5$).

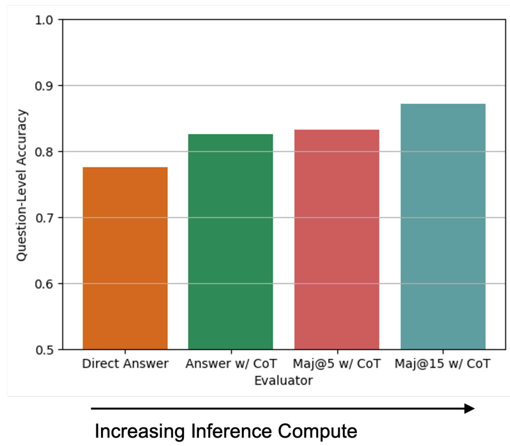


Figure 3: Question-level accuracy of GPT-4o checklist answers on Internal.

We follow Qin et al. (2024) and use the Pairwise Label Distance (PLD) to measure agreement between TICK and human preference labels. PLD is a metric designed to capture the intuition that predicting a win as a tie is not as bad as predicting a win as a loss. It takes a label and a prediction, and produces a value in $\{0, 1, 2\}$. A PLD of 0 indicates the exact match of a preference label (win, loss or tie), (i.e., PLD-0 is equivalent to label accuracy in this setup). A PLD of 1 implies a misclassification in scenarios where the ground truth label was a tie. A PLD of 2 corresponds to an inverted preference relative to the human preference label (e.g., predicting loss when the ground truth label is win). The Weighted Pairwise Label Distance (WPLD) is then defined as $\text{WPLD} = \sum_{i=0}^{2} \frac{i}{N} \sum_{j=0}^{N} \mathbb{I}[\text{PLD}_j = i]$, where N is the number of instructions. The WPLD thus ranges from 0 – 2, with a lower value indicating stronger agreement.

We compare making preference judgments via TICK against directly prompting the judge LLM to express a preference (Preference) and scoring each response individually (Direct Scoring). For direct scoring, we prompt the judge LLM to produce a 1 – 5 score for each response, and we say the higher scoring response is preferred. We also include a hybrid of TICK and direct scoring (Check-then-Score), where checklists are included in the judge prompt, but judges are not required to explicitly answer each checklist question, similar to how curated checklists are used in WildBench Lin et al. (2024). We have the judge LLM use CoT in all cases, but do not use majority voting. Response pairs are formed out of generations from Command-R+, GPT-4o and Claude-3-Sonnet (Anthropic, 2023). We use GPT-4o as the judge LLM.

In Table 1, we see that TICK agrees most strongly with human preferences, in terms of achieving the lowest overall WPLD. TICK is also the only LLM-as-judge evaluation to achieve a PLD of 0 more often than not. Check-then-score also agrees more strongly with humans than direct scoring, which confirms the general utility of checklists in evaluation. However, the fact that Check-the-Score still lags behind TICK provides evidence that explicitly answering and aggregating checklist answers is necessary to fully utilise checklists. Despite the fact that preference judgements are very common, prompting the judge to directly produce a preference produces low agreement with humans. Overall, these results show that *LLM-as-judge evaluations benefit from a more precisely structured and granular scoring protocol*, even when that protocol is task-agnostic and generally applicable, as in the case of TICK.

D Prompt Templates

D.1 Checklist Generation

```
Please help judge an AI assistant's response to an instruction by
providing an evaluation checklist.
To write a specific evaluation checklist, you get given the
following entity each time:
INSTRUCTION: An instruction that has been given to an AI
assistant.

## Task Details
Your task is to come up with an evaluation checklist list for a
given INSTRUCTION.
This evaluation checklist should be a list of questions that ask
whether or not specific criteria relevant to the INSTRUCTION were
met by an AI assistant's response.
Criteria covered by your checklist could be explicitly stated in
the INSTRUCTION, or be generally sensible criteria for the
problem domain.
You should, however, try to be concise and not include
unnecessary entries in your checklist.

Checklist questions should:
- **Be answerable by 'yes' or 'no'**, with 'yes' meaning that the
response successfully met the corresponding requirement.
- **Be comprehensive, but concise**, meaning that all criteria
directly relevant to the INSTRUCTION should be represented by a
question, but only questions that are very clearly relevant
should be included.
- **Be precise**, meaning that checklist questions should avoid
vague wording and evaluate specific aspects of a response,
directly using the phrasing of the INSTRUCTION where appropriate.

You should always analyse the INSTRUCTION before providing an
evaluation checklist.

## Response Format
Analysis: xxx
Answer: CHECKLIST QUESTIONS (each question should appear on a new
line)

## Examples
{examples}

## Real Task

### INSTRUCTION
{message}

### Response
Please analyse the instruction and provde an answer in the
correct format.
Remember that each question should be phrased such that answering
with 'yes' would mean that the response **successfully**
fulfilled the criteria being assessed by the question.
In most cases, your checklist should contain at least two
questions, but no more than eight.
```

D.2 Checklist Evaluation

This prompt is adapted from (Qin et al., 2024).

```
Please act as a fair judge. Based on the provided Instruction and
Generated Text, analyse the Generated Text and answer the
Question that follows with 'YES' or 'NO'.
Your selection should be based on your judgment as well as the
following rules:

- YES: Select 'YES' if the generated text entirely fulfills the
condition specified in the question. However, note that even
minor inaccuracies exclude the text from receiving a 'YES'
rating. As an illustration, consider a question that asks, ``Does
each sentence in the generated text use a second person?'' If
even one sentence does not use the second person, the answer
should NOT be `YES'. To qualify for a 'YES' rating, the generated
text must be entirely accurate and relevant to the question.

- NO: Opt for 'NO' if the generated text fails to meet the
question's requirements or provides no information that could be
utilized to answer the question. For instance, if the question
asks, 'Is the second sentence in the generated text a compound
sentence?' and the generated text only has one sentence, it
offers no relevant information to answer the question.
Consequently, the answer should be `NO'.

## Output Format
Analysis: xxx
Answer: YES / NO (this should be either 'YES' or 'NO')

## Evaluation Information

**Instruction**
{message}

**Generated Text**
{generation}

**Question**
{question}

Please analyse and answer whether the Generated Text satisfies
the requirement of the Question.
```

D.3 Preference

```
Please act as a fair judge. Based on the provided Instruction and
Responses, analyse the Responses and provide a preference.
Your selection should be based on your judgment and correspond to
one of the following preference rankings:

1. Response A is better than Response B.
2. Response A and Response B are near-identical.
3. Response B is better than Response A.

The 'near-identical' option (i.e., option 2) should be chosen
only if the differences between the two responses are
semantically and syntactically insignificant, such as 'The
```

```
correct answer is New York' and 'The right answer is New York'.
In other words, if the two responses are substantially different
in terms of their content, you must identify a preference for one
of the responses. **Responses that are different in content
but similar in quality are NOT near-identical.**

## Output Format
Analysis: xxx
Answer: PREFERENCE RANKING (this should be an integer from 1-3
and nothing else)

## Evaluation Information

**Instruction**
{message}

**Response A**
{generation_1}

**Response B**
{generation_2}

Please analyse the Responses and provide a preference ranking (1,
2, or 3). Remember to stick to the requested Output Format.
```

D.4 Direct Scoring

```
Please act as a fair judge. Based on the provided Instruction and
Generated Text, analyse the Generated Text and provide a 1-5
integer score.
Your selection should be based on your judgment as well as the
following guidelines for each possible score:

1. Horrible: The Generated Text is unintelligibly written
(incomplete sentences, leaps in logic, flagrant mechanical
errors) or has majorly incorrect or unverifiable information.
2. Bad: The Generated Text is occasionally difficult to
understand, dotted with minor factual or mechanical errors, or
missing crucial formatting elements.
3. Okay: The Generated Text expresses useful information, is
readable, has no factual errors, and has no more than a minor
mechanical error or two. Though it may be informative to those
unfamiliar with the subject matter, it is not overly insightful,
engaging, or likely to hold up to expert scrutiny.
4. Great: The Generated Text clearly expresses useful information
at an expert level, is readable, and has no factual or mechanical
errors. It could just use a quick adjustment with tone or length.
5. Excellent: The Generated Text clearly expresses useful
information at an expert level, is readable, has no factual or
mechanical errors, and is the perfect length and tone with regard
to the prompt.

## Output Format
Analysis: xxx
Answer: SCORE (this should be an integer from 1-5 and nothing
else)

## Evaluation Information
```

```
**Instruction**
{message}

**Generated Text**
{generation}

Please analyse the Generated Text and provide a 1-5 integer score
according to the guidelines. Remember to stick to the requested
Output Format.
```

D.5 Check-then-Score

```
Please act as a fair judge. Based on the provided Instruction and
Generated Text, analyse the Generated Text and provide a 1-5
integer score.
You will also be provided with a Checklist that should help to
inform your selection.
Your selection should be based on your judgment as well as the
following guidelines for each possible score:

1. Horrible: The Generated Text is unintelligibly written
(incomplete sentences, leaps in logic, flagrant mechanical
errors) or has majorly incorrect or unverifiable information.
2. Bad: The Generated Text is occasionally difficult to
understand, dotted with minor factual or mechanical errors, or
missing crucial formatting elements.
3. Okay: The Generated Text expresses useful information, is
readable, has no factual errors, and has no more than a minor
mechanical error or two. Though it may be informative to those
unfamiliar with the subject matter, it is not overly insightful,
engaging, or likely to hold up to expert scrutiny.
4. Great: The Generated Text clearly expresses useful information
at an expert level, is readable, and has no factual or mechanical
errors. It could just use a quick adjustment with tone or length.
5. Excellent: The Generated Text clearly expresses useful
information at an expert level, is readable, has no factual or
mechanical errors, and is the perfect length and tone with regard
to the prompt.

## Output Format
Analysis: xxx
Answer: SCORE (this should be an integer from 1-5 and nothing
else)

## Evaluation Information

**Instruction**
{message}

**Generated Text**
{generation}

**Checklist**
Use this checklist to guide your evaluation, but do not limit
your assessment to the checklist.
{checklist}
```

```
Please analyse the Generated Text and provide a 1-5 integer score
according to the guidelines. Remember to stick to the requested
Output Format.
```

D.6 Self-Refinement

Using Self-TICK as Critiques

```
Please use the feedback provided below to improve your previous
response to an instruction.
You will be given the following entities:
- INSTRUCTION: An instruction that has been given to an
assistant.
- RESPONSE: Your previous response.
- FEEDBACK: A list of 'yes'/'no' questions about the response and
their answers. An answer of 'yes' corresponds to a pass for that
question and an answer of 'no' correspnods to a fail.

## Task Description
Your task is to improve the RESPONSE to the INSTRUCTION based on
the FEEDBACK. You should try to address any 'no' answers in the
feedback whilst maintaining any 'yes' answers.
If all answers in feedback are 'yes', simply respond with your
original RESPONSE.
Provide a plan to improve the RESPONSE based on the INSTRUCTION
and FEEDBACK and then rewrite the RESPONSE with your
improvements.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

**FEEDBACK**
{feedback}

## Response Format (IMPORTANT)
Plan: xxx
Answer: NEW RESPONSE
After saying 'Answer: ' you must say nothing else besides the
improved answer.

Now please plan and write a new RESPONSE, based on the
INSTRUCTION and FEEDBACK.
```

Gathering Unstructured Self-Critiques

```
Please analyse a response to a particular instruction and provide
feedback on how the response can be improved.
You will be given the following entities:
- INSTRUCTION: An instruction that has been given to an
assistant.
- RESPONSE: Your previous response.

## Task Description
Your task is to provide feedback that will help improve the
```

```
RESPONSE to the INSTRUCTION.
Please analyse the RESPONSE and provide your critical feedback,
pointing to specific actionable improvements that can be made.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

Now please provide your feedback.
```

Using Unstructured Self-Critiques

```
Please use the feedback provided below to improve your previous
response to an instruction.
You will be given the following entities:
- INSTRUCTION: An instruction that has been given to an
assistant.
- RESPONSE: Your previous response.
- FEEDBACK: Feedback on your previous response.

## Task Description
Your task is to improve the RESPONSE to the INSTRUCTION based on
the FEEDBACK.
Provide a plan to improve the RESPONSE based on the INSTRUCTION
and FEEDBACK and then rewrite the RESPONSE with your
improvements.

## Information
**INSTRUCTION**
{message}

**RESPONSE**
{response}

**FEEDBACK**
{feedback}

## Response Format (IMPORTANT)
Plan: xxx
Answer: NEW RESPONSE
After saying 'Answer: ' you must say nothing else besides the
improved RESPONSE.
The new RESPONSE must exactly match the formatting of the
original.

Now please plan and write a new RESPONSE, based on the
INSTRUCTION and FEEDBACK.
```

E Human Annotation Details

The same pool of trained annotators was used in all human annotation processes. The training undergone by annotators includes general, task-agnostic training covering high level guidelines for annotating the outputs of an AI assistant, as well as task-specific instructions and examples. At all stages in the annotation process, we are able to interact with annotators to answer any questions and respond to requests for clarification.

E.1 Checklist Writing

The following instructions are given to annotators. Some sections are paraphrased for brevity.

```
Large language models are trained to respond to user
instructions, which can often be complex. To best evaluate
responses to a set of user instructions, we're exploring the
viability of writing custom queries for each prompt to determine
which aspects of a complex instruction were followed correctly
and which were not.

In this project, you will write checklists of questions for given
instructions that ask whether each aspect of an instruction was
met by a model output.

## Key Concept: Facet

### Definition

Facets are distinct, individual elements of a prompt,
corresponding to the capabilities and constraints that the model
output should meet.

## Tips and Tricks

- **Checklist questions must be answerable by either** 'yes'
**or** 'no'**.** The idea is that the questions are to be asked
of responses to the given prompt; 'yes' means that the response
fulfilled the facet of the instruction, and 'no' means it did
not.
- **Ensure each facet you identify is represented by a single
question.** This means you can provide as many questions for a
prompt as you see fit for that prompt, but questions should
overlap as little as possible (i.e., there should not be many
questions addressing the same facet).
- **Questions should be as specific or unspecific as the facet
they correspond to.** For example:
    Prompt 1: Write a short paragraph.
    Q1: Is the paragraph short?
    Prompt 2: Write a 3 sentence paragraph.
    Q2: Is the paragraph 3 sentences long?

## Examples
{examples}

## Task Instructions
1. Carefully analyse the provided prompt.
2. Label the task:
    a)Answer the two questions about the prompt being safe and
    making sense.
    b) Unless answering 'no' to one of the above, write a list of
    checklist questions that could be asked of any response to
    the prompt.
3. Submit the task. Great work!
```

E.2 Preference Labelling

The following instructions are given to annotators. Some sections are paraphrased for brevity. These instructions are to be interpreted in the context of a more comprehensive *model output annotation style guide* that is not included here.

```
In this project, you will be indicating your preference between
two responses to a single prompt generated by two different
LLM-powered chatbots. Evaluating responses may involve making
trade-offs between several criteria. You should do your best to
navigate these trade-offs depending on the task.

Given an instruction and two responses, please indicate which
response you prefer on the following sliding scale:
1. Response A is much better than Response B.
2. Response A is better than Response B.
3. Response A and Response B are near-identical.
4. Response B is better than Response A.
5. Response B is much better than Response A.

The ’near-identical’ option (i.e., option 2) should be chosen
only if the differences between the two responses are
semantically and syntactically insignificant, such as ’The
correct answer is New York’ and ’The right answer is New York’.
In other words, if the two responses are substantially different
in terms of their content, you must identify a preference for one
of the responses. **Responses that are different in content
but similar in quality are NOT near-identical.**
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The experimental results presented in the paper validate all claims made in the abstract and introduction.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: This discussion can be found in the Appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: No theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: All prompts (see Appendix), methods and models are used are detailed in the paper. Publicly available datasets are used for STICK experiments and we will be releasing the Internal dataset used for the human-model agreement experiment.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: We have not yet made our code publicly available, but will be doing so before the time of the conference.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: All experimental details are included in the paper.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: Due to the cost of LLM API calls, all experiments were only run with a single seed.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: Experiments for this paper relied solely on API access to LLMs. We did not perform any model training or local hosting for inference.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work strictly adheres to the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The potential societal implications of having LLMs judge their own outputs, without human evaluation, is mentioned in the limitations section of the Appendix.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: No misuse risks posed by this work.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All models and datasets used are properly credited, with licensing terms respected.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets released.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [Yes]

    Justification: Details of human annotation setup and instructions provided in the Appendix.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [No]

    Justification: Internal safety guidelines and our annotation operations team act as a form of internal ethics review.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.