Literature Review: Evaluating Large Language Models as a Judge in Code Generation Tasks

Tim Boleslawsky

I. Introduction

Large Language Models (LLMs) have become increasingly central to automated code generation and evaluation tasks. Their ability to both produce solutions and assess responses has opened new research avenues, but also raised questions about reliability, bias, and alignment with human judgment. A crucial research strand focuses on *LLMs as a judge*, where a model evaluates whether a generated response meets some qualitative criteria or the requirements of an input request. This literature review surveys the current status of research concerning the concept of LLM as a judge.

II. LLMs as Judges: Conceptual Background

The paradigm of *LLMs as a judge* was crystallized by Li et al. [2024], who introduced JudgeLM as a framework for studying how models can function as evaluators. The central motivation is to reduce reliance on costly human annotations and to create scalable evaluation pipelines for tasks like instruction-following, dialogue quality, and code correctness. JudgeLM and subsequent works emphasize that a judge must do more than measure surface similarity—it must capture nuanced requirement satisfaction and correctness.

The problem is especially pronounced in code generation. Unlike open-ended text generation, code tasks usually have well-defined correctness criteria. Still, verifying alignment between requirements and generated code often goes beyond syntactic validity: it involves checking logical correctness, adherence to constraints, and robustness.

III. Datasets for Evaluating LLM-as-Judge Frameworks

Datasets play a critical role in advancing this line of research. Several benchmark collections have emerged:

- **MT-Bench and Chatbot Arena**: Introduced by Zheng et al. [2023], these datasets capture multi-turn dialogue and human preferences, providing ground truth for comparison between human and model judgments.
- **Eval4NLP Shared Tasks**: These benchmarks focus on automatic evaluation methods in NLP, highlighting the challenge of aligning system-based judgments with human annotations [**?**]. While not specific to code, they illustrate methods transferable to code evaluation.
- **Code Evaluation Benchmarks**: Works such as Chen et al. [2021] and **?** introduced HumanEval and APPS, where correctness is tested via unit tests. However, Fu et al. [2023] stress that correctness is only part of the picture: requirement satisfaction and style adherence are equally important.
- **Multi-judge Datasets**: **?** collected data where multiple judges (human and LLMs) evaluated outputs. This allowed meta-evaluation—comparing how well judges themselves align with majority or expert judgments.

These datasets underpin both direct evaluation (testing whether a response is correct) and meta-evaluation (testing whether an evaluator is reliable).

IV. Evaluation Metrics

Metrics for evaluating LLMs as judges fall into three categories:

A. Agreement with Human Judgment

The most direct measure of an evaluator's quality is agreement with human annotators.Fu et al. [2023] introduced GPTScore, which correlates LLM-based evaluation with human preference judgments across text and code tasks. Similarly, **?** stress that correlation coefficients such as Kendall's tau and Spearman's rho are widely adopted to measure alignment.

B. Functional Correctness

For code generation, execution-based evaluation remains critical. HumanEval [Chen et al., 2021] defines correctness as passing predefined test cases. However, execution-only approaches are limited because they miss requirement-specific constraints not encoded in tests. Thus, judge models are increasingly combined with functional correctness tests to balance efficiency and comprehensiveness.

C. Robustness and Bias Metrics

Recent works emphasize robustness of LLM judges.**?** propose adversarial settings where LLM judges must handle misleading or ambiguous instructions. They also explore judge consistency across paraphrased prompts and semantically equivalent but syntactically different

outputs. Bias metrics, such as preference for verbosity or syntactic patterns, are being proposed to better characterize weaknesses of evaluators.

V. Evaluating the Evaluator

A particularly novel contribution of this field is the explicit *evaluation of evaluators*. Unlike classical metrics, which assume the metric itself is ground truth, here researchers directly assess judge reliability.

Fu et al. [2023] demonstrate that GPT-4-based judges often outperform traditional automatic metrics in correlation with human ratings, but they also show systematic biases (e.g., preferring longer outputs). Similarly, **?** highlight that different LLM judges can disagree substantially, motivating ensemble or consensus-based evaluation frameworks.

Another perspective is *calibration*. A well-calibrated judge should not only predict correctness but also assign confidence scores that reflect uncertainty. Early works suggest LLM judges often overstate confidence [Zheng et al., 2023].

VI. Open Challenges

Despite progress, several challenges remain:

- **Dependence on Human Labels**: While LLM judges reduce annotation costs, they still rely on high-quality labeled datasets for benchmarking.
- **Generalization Across Domains**: Judges trained on dialogue or summarization tasks may not generalize well to code evaluation.
- **Transparency and Bias**: Understanding biases in judge models remains limited. Future work must identify and mitigate systematic errors.
- **Multi-level Evaluation**: Capturing both functional correctness and requirement adherence is still unsolved for complex multi-step tasks.

VII. Conclusion

The literature on LLMs as judges highlights a rapidly evolving research area with deep implications for automated evaluation. From JudgeLM to GPTScore, scholars have begun to systematize evaluation frameworks that balance correctness, alignment with human preferences, and robustness. While progress is evident, meta-evaluation—the evaluation of evaluators—remains both a conceptual and technical challenge. Advancing this area will require larger multi-domain datasets, robust bias analysis, and frameworks that integrate human and automated evaluation seamlessly.

References

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William H. Guss, Alex Nichol, Igor Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374. HumanEval dataset introduced; DOI: 10.48550/arXiv.2107.03374.

Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire. *arXiv preprint arXiv:2302.04166*, 2023. URL https://arxiv.org/abs/2302.04166.

Haitao Li, Qian Dong, Junjie Chen, Huixue Su, Yujia Zhou, Qingyao Ai, Ziyi Ye, and Yiqun Liu. Llms-as-judges: A comprehensive survey on llm-based evaluation methods. *arXiv preprint arXiv:2412.05579*, 2024. URL https://arxiv.org/abs/2412.05579.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena. *arXiv preprint arXiv:2306.05685*, 2023. URL https://arxiv.org/abs/2306.05685.