Uncovering Effective Steering Strategies in Code-Generating LLMs with Socratic Feedback

ZHENG ZHANG*, University of Notre Dame, USA
ALEX C. WILLIAMS, AWS AI, Amazon, USA
JONATHAN BUCK, AWS AI, Amazon, USA
XIAOPENG LI, AWS AI, Amazon, USA
MATTHEW LEASE, AWS AI, Amazon, USA
LI ERRAN LI, AWS AI, Amazon, USA

Large Language Models (LLMs) are rapidly becoming commonplace tools for generating code solutions for complex programming problems. Alongside their widespread availability, the adoption of code-generating LLMs has been driven by a myriad of code-related features (e.g., self-debugging) that empower their ability to generate functionally correct code. Despite these recent advances, LLMs remain largely incapable of solving many programming problems of significant complexity unless otherwise steered through human feedback. In this paper, we uncover the strategies that professional programmers employ in steering code-generating LLMs toward successful outcomes. We present findings from a study in which 38 professional programmers used natural language to steer GPT-4 in generating solutions for programming problems of varying difficulty. We motivate our study design and analysis through the lens of *Socratic Feedback*, an inversion of the popular *Socratic Questioning* paradigm that frames professional programmers as "model teachers" to code-generating LLMs. Through an analysis of 80 conversational transcripts from interactions with GPT-4, we map observed error-feedback pairs to four conceptual stages of interactive steering in the context of code generation: *Understanding*, *Planning*, *Implementation*, and *Testing*. We ground our data analysis in these four stages and conclude by discussing their implications for improving the performance of code-generating LLMs through combined lens of user experience and functional utility.

CCS Concepts: • **Human-centered computing → Empirical studies in HCI**.

Additional Key Words and Phrases: Large language models, code generation, steering.

ACM Reference Format:
Zheng Zhang, Alex C. Williams, Jonathan Buck, Xiaopeng Li, Matthew Lease, and Li Erran Li. 2024. Uncovering Effective Steering Strategies in Code-Generating LLMs with Socratic Feedback. In *CSCW'24 Workshop on Supporting User Engagement in Testing, Auditing, and Contesting AI, October 15, 2023, Minneapolis, MN, USA*. ACM, New York, NY, USA, 2 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Recent advances in natural language processing (NLP) and automated code generation have combined to drive the development of Large Language Models (LLMs) that can generate code via natural language prompts. Empirical studies suggest that LLMs can provide significant gains to productivity and has transformed practices of daily programming. Programmers have frequently relied on those publicly available LLMs (e.g. ChatGPT [5]) and coding assistants (e,g. Copilot [2]) for code suggestion and brainstorming in their work and study.

Nonetheless, code-generating LLMs reportedly still struggle with intricate coding challenges that require a deep understanding of the problem, task breakdown, and the nuanced application of algorithms and libraries within given specification [1, 4, 6]. Recently, some advancements have revealed that newer LLMs possess a *self-debugging* ability [3, 6].

*Work completed during an internship at Amazon.

This lets them enhance code generation by iteratively analyzing errors from previous program attempts based on unit test outcomes, akin to common trial-and-error strategy used by human programmers. Yet, entirely relying on self-debugging capability comes with two major vulnerabilities. First, the model might erroneously pinpoint underlying reason for code failure. Moreover, even if the model correctly identifies the failure, it may struggle with generating contingent feedback to effectively steer the following code refinement. These obstacles account for the modest performance improvements achieved by self-debugging frameworks in the context of complex programming tasks, such as LeetCode's medium and hard-level problems [3, 6].

In this paper, we conducted an empirical study that understands how professional programmers could steer a self-debug capable model, GPT-4[1], to generate functionally correct code for programming problems that it failed to solve alone. Specifically, we encouraged programmers to employ the Socratic feedback approach that is commonly used in argumentation and tutoring. Instead of directly providing answers, this approach leverages targeted questions or prompts to ignite critical thinking and empower learners to formulate their own solutions. This is akin to the dynamic in college programming tutoring sessions, where the instructor offers incremental feedback corresponding to the learner's most recent attempt. Meanwhile, the learner is required to undergo several rounds of debugging efforts before seeking additional feedback. Our study with 38 professional programmers found that GPT-4 is able to solve originally failed competition-level programming problems with a few rounds of human Socratic feedback. Moreover, through programmers' self-annotations on the conversation log data, we uncovered a variety of error types that GPT-4 committed during both the instruction-following and self-debugging phases. We also identified a group of Socratic feedback techniques that programmers employed to guide the Language Learning Model (LLM). Furthermore, we analyzed programmers' challenges in steering GPT-4 for coding tasks.

REFERENCES

[1] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. 2021. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732* (2021).

[2] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[3] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).

[4] Hantian Ding, Varun Kumar, Yuchen Tian, Zijian Wang, Rob Kwiatkowski, Xiaopeng Li, Murali Krishna Ramanathan, Baishakhi Ray, Parminder Bhatia, Sudipta Sengupta, et al. 2023. A static evaluation of code completion by large language models. *arXiv preprint arXiv:2306.03203* (2023).

[5] 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 (2022), 27730–27744.

[6] Noah Shinn, Beck Labash, and Ashwin Gopinath. 2023. Reflexion: an autonomous agent with dynamic memory and self-reflection. *arXiv preprint arXiv:2303.11366* (2023).

[1]We used Advanced Data Analysis plugin of GPT-4