Self-Invoking Code Generation and Robustness to Distribution Shifts in Programming Tasks

Assignee Research

June 9, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: Does the self-invoking code generation paradigm enhance robustness against distribution shifts in code generation tasks compared to single-turn generation baselines. 12 claims were extracted from source literature; 2 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 4.8/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: HumanEval Pro and MBPP Pro: Evaluating Large Language Models on Self-invoking Code Generation. Research question: Does the self-invoking code generation paradigm enhance robustness against distribution shifts in code generation tasks compared to single-turn generation baselines?.

2 Methodology

Systematic literature search across multiple databases yielded 14 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 4.8/10.

3 Results

14 papers retrieved. 12 claims extracted; 2 independently verified. Quality review score: 4.8/10.

4 Limitations

This report is a machine-generated literature synthesis and does not constitute original research. Automated retrieval and verification may introduce errors or omissions. Review scores reflect automated assessment, not human peer review. Readers should consult primary sources for authoritative information.

5 Extracted Claims

Claim	Verified	Confidence
o1-mini achieves 96.2% pass@1 on HumanEval but only 76.2% on HumanEval Pro.	✓	0.25
Instruction-tuned models are less efficient on self-invoking code generation than traditional code generation tasks.	✓	0.30
HumanEval and MBPP serve as fundamental benchmarks, focusing on Python function completion tasks with test-driven evalua	×	0.11
Several benchmarks have expanded code evaluation benchmarks to encompass multiple programming languages, complex tasks l	×	0.11
The benchmark construction process involves three steps: Self-invoking problem Generation, Solutions Generation, and Tes	×	0.10
Deepseek-V2.5 is used to generate self-invoking problems, candidate solutions, and test inputs.	×	0.07
An iterative method involving Python execution check and manual review is employed to ensure that all test cases pass su	×	0.02
The final execution results are used to construct complete test cases with assert command.	×	0.02
Qwen2.5-Coder-7B-base achieves 59.6% on HumanEval Pro and 38.6% on MBPP Pro.	×	0.12
Qwen2.5-Coder-7B-instruct achieves 64.9% on HumanEval Pro and 35.1% on MBPP Pro.	×	0.09
DeepseekCoder-33B-base achieves 71.9% on HumanEval Pro and 38.6% on MBPP Pro.	×	0.13
DeepseekCoder-33B-instruct achieves 80.7% on HumanEval Pro and 43.9% on MBPP Pro.	×	0.10

References

- http://arxiv.org/abs/2502.20380v2
- http://arxiv.org/abs/2507.10646v5
- http://arxiv.org/abs/2412.21199v2