LiveCodeBench Performance Analysis of Competitive Programming Language Models

Assignee Research

June 3, 2026

Abstract

This report synthesises findings from 15 peer-reviewed papers addressing the following research question: LiveCodeBench competitive programming language model performance analysis. 20 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 3.9/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: LiveCodeBench: Holistic and Contamination Free Evaluation of Large Language Models for Code. Research question: LiveCodeBench competitive programming language model performance analysis.

2 Methodology

Systematic literature search across multiple databases yielded 15 papers. Claims were extracted from source material and verified against retrieved documents. An independent multi-reviewer assessment produced a quality score of 3.9/10.

3 Results

15 papers retrieved. 20 claims extracted; 0 independently verified. Quality review score: 3.9/10.

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
LiveCodeBench problems are curated from LeetCode, AtCoder, and CodeForces.	×	0.13
The source websites periodically host contests containing problems that assess coding and problem-solving skills.	×	0.03
Source problems consist of a natural language problem statement and example input-output examples.	×	0.11
The goal of source problems is to write a program that passes a set of hidden tests.	×	0.03
Thousands of participants solve problems on the source websites, vetting them for clarity and correctness.	×	0.01
HTML scrapers were written to collect problems and metadata from LeetCode, AtCoder, and CodeForces.	×	0.07
Mathematical formulas in the collected problems are parsed.	×	0.02
Problems containing images are excluded from the dataset.	×	0.05
Problems not suitable for grading by input-output examples, such as those accepting multiple correct answers, are exclud	×	0.07
Problems requiring the construction of data structures are excluded from the dataset.	×	0.05
Ground truth solutions and test cases are collected whenever directly available on the source platforms.	×	0.05
Each problem in the dataset is stored as a tuple containing a natural language problem statement (P), test cases (T), an	×	0.06
A contest date (D) is associated with each problem to mark its release date.	×	0.04
The dataset supports filtering problems based on whether their release date falls within specific time windows.	×	0.03
Model performance can be measured on benchmark problems released after the model's training cutoff date.	×	0.06
A UI was developed to allow comparing models on problems released during different time windows.	×	0.04
Tests are collected from platform websites whenever possible.	×	0.02
When platform tests are unavailable, GPT-4-Turbo is used to generate tests following the approach of Liu et al. (2023b).	×	0.02
The test generation approach constructs generators that sample inputs based on problem specifications using in-context l	×	0.04

References

- http://arxiv.org/abs/2403.03788v1
- http://arxiv.org/abs/2403.07974v2
- http://arxiv.org/abs/1711.03620v1