Is Your Code Generated by ChatGPT Really Correct?
Rigorous Evaluation of Large Language Models for Code Generation

Jiawei Liu[I*] Chunqiu Steven Xia[I*] Yuyao Wang[N] Lingming Zhang[I]

University of Illinois Urbana-Champaign[I] Nanjing University[N] *Co-primary

TL;DR Evaluating LLM-generated code over code synthesis benchmarks with "3 test-cases" is **NOT** enough!

Introduction

1. Program synthesis is a long-standing problem in computer science
2. LLM-based codegen shows great promise to synthesize code following user intent
3. Existing programming benchmarks are manually constructed with small number of test-cases in order to measure functional correctness

Is the code generated by LLMs **really correct**?

❌ `common([6,8,1], [6,8,1]) ⇒ [8,1,6]`

```
def common(l1: list, l2: list) → list:
  """Return sorted unique common elements for two lists"""
  common_elems = list(set(l1).intersection(set(l2)))
  common_elems.sort()
  return list(set(common_elems))
```

generated by ***ChatGPT***

`list`

`set` is *unordered*! `list→set` is NOT order-preserving!

✅ `common([4,3,2,8], [...`
✅ `common([5,3,2,8], [3,2]) ⇒ [2,3]`

Fig. 1. ChatGPT-generated *wrong* code was considered correct by HumanEval

Existing code benchmarks (*e.g.,* HumanEval) heavily rely on manually constructed test-cases to evaluate LLM solutions. However, crafting high-quality tests is laborious and as such, current programming benchmarks are inadequate for assessing the actual correctness of LLM-generated code with the following limitations:

Insufficient testing. Current programming benchmarks often only include < 10 test-cases per problem. Furthermore, these test-cases are too simply to fully explore the functionality of the code or cover corner cases. As such, logically flawed solutions can still pass all tests and be *misconsidered* as correct by existing benchmarks.

Imprecise problem description. Natural language description of the problem is used as the input for code generation. However, these task descriptions in existing benchmarks are oftentimes too vague to fully clarify the expected program behaviours. As such, different LLMs can interpret the problem differently, leading to capable LLMs misjudged as incapable.

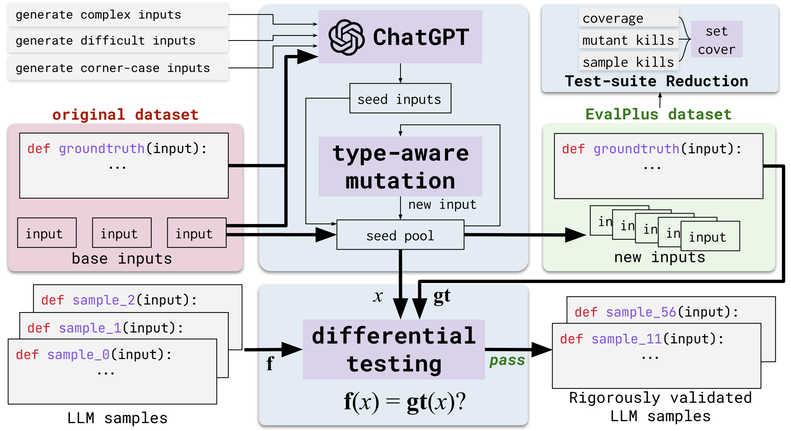
Can we *rigorously* evaluate the functional correctness?

We propose: EvalPlus – an evaluation framework to augment code benchmarks with additional tests for precisely evaluating the correctness of LLM-generated code.

We produce: HumanEval$^+$ – an extended version of HumanEval with $80\times$ more tests and perform extensive evaluation on over 26 popular LLMs.

We found: that pass@k on the new and more rigorous dataset is regularized to up-to 19.3-28.9% lower than the base HumanEval.

Approach



Test Input Creation

EvalPlus automatically generates a set of high-quality test-cases to rigorously evaluate the functional correctness of LLM-generated code.

Seed initialization via ChatGPT EvalPlus first construct prompt using *(i)* the ground-truth solution of the problem; *(ii)* a set of test inputs as demonstration; and *(iii)* an instruction to encourage ChatGPT to come up with interesting inputs.

Type-aware input mutation. EvalPlus then continuously mutates existing inputs using mutation operators by observing the specific input types. Through this process, EvalPlus can efficiently obtain a large set of high-quality test-cases.

Test-suite reduction EvalPlus can also perform test-suite reduction using a variety of requirements (*i.e.,* coverage, mutant and sampling killings) to find a minimal subset of test-cases to both ensure effectiveness and reduce testing time.

Program Input Contracts

To ensure generated test-cases do not invoke any invalid program behaviors, we systematically add input contracts in the form of code assertions. Using program contracts, EvalPlus directly filters out any invalid test inputs.

HumanEval$^+$

EvalPlus transforms HumanEval into **HumanEval$^+$** by adding **80x** unique test-cases and also fixing any incorrect ground-truth solutions.

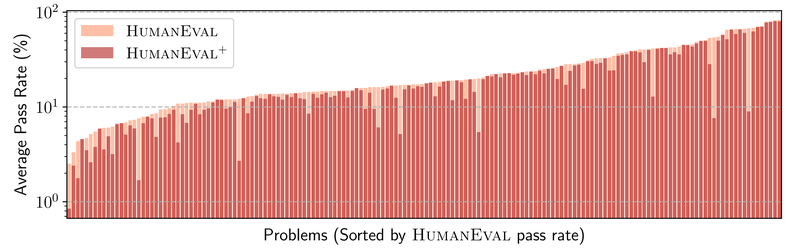
	#Tests				#Tasks
	Avg.	Medium	Min.	Max.	
HumanEval	9.6	7.0	1	105	
HumanEval$^+$	764.1	982.5	12	1,100	164
HumanEval$^+$-mini	16.1	13.0	5	110	

Result Highlights

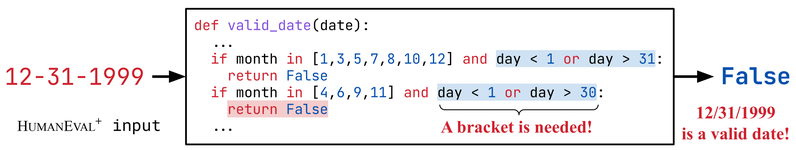
Evaluation using HumanEval$^+$. We evaluate 26 popular and state-of-the-art LLMs (*e.g.,* GPT-4, ChatGPT, and CodeLlama) on HumanEval$^+$.

- Almost all pass@k **consistently drop** compared to using the base benchmark.
- Performance drop is significant with up-to **23.1%** (greedy) / **19.3%** (pass@1) / **24.9%** (pass@10) / **28.9%** (pass@100) reduction over the evaluated models.
- Both WizardCoder-CodeLlama and Phind-CodeLlama outperform ChatGPT on HumanEval$^+$, while none of them could on HumanEval.

Pass rate distribution. Not all HumanEval problems see an equal amount of drop in pass rate. We observe that **multi-condition and reasoning problems** in particular are the most difficult for LLMs to solve.



Incorrect ground-truth in HumanEval. We also found **18** defects (**11%** of problems) even in the original ground-truth in HumanEval. Categorized into *(i)* Unhandled edge-case; *(ii)* Bad logic; and *(iii)* Performance issue.



Try It Out

EvalPlus is available on PyPI (`evalplus`) and GitHub (`evalplus/evalplus`)

So ... who is the best LLM coder? Take a look at the EvalPlus leaderboard: **evalplus.github.io/leaderboard.html**

paper *leaderboard* *code*

✉ {jiawei6, chunqiu2}@illinois.edu