What can Large Language Models Capture about Code Functional Equivalence?

Nickil Maveli Antonio Vergari Shay Cohen

University of Edinburgh

TL;DR

The performance gap between match-based and LLM-based metrics in assessing the functional equivalence of code snippets is **minimal**, with both approaches showing a concerning lack of depth in understanding code semantics.

Motivation

Determining if two programs are functionally equivalent is fundamentally **undecidable**. Consequently, existing code benchmarks depend on manually created test cases to assess LLM performance. This execution-based method has several drawbacks.

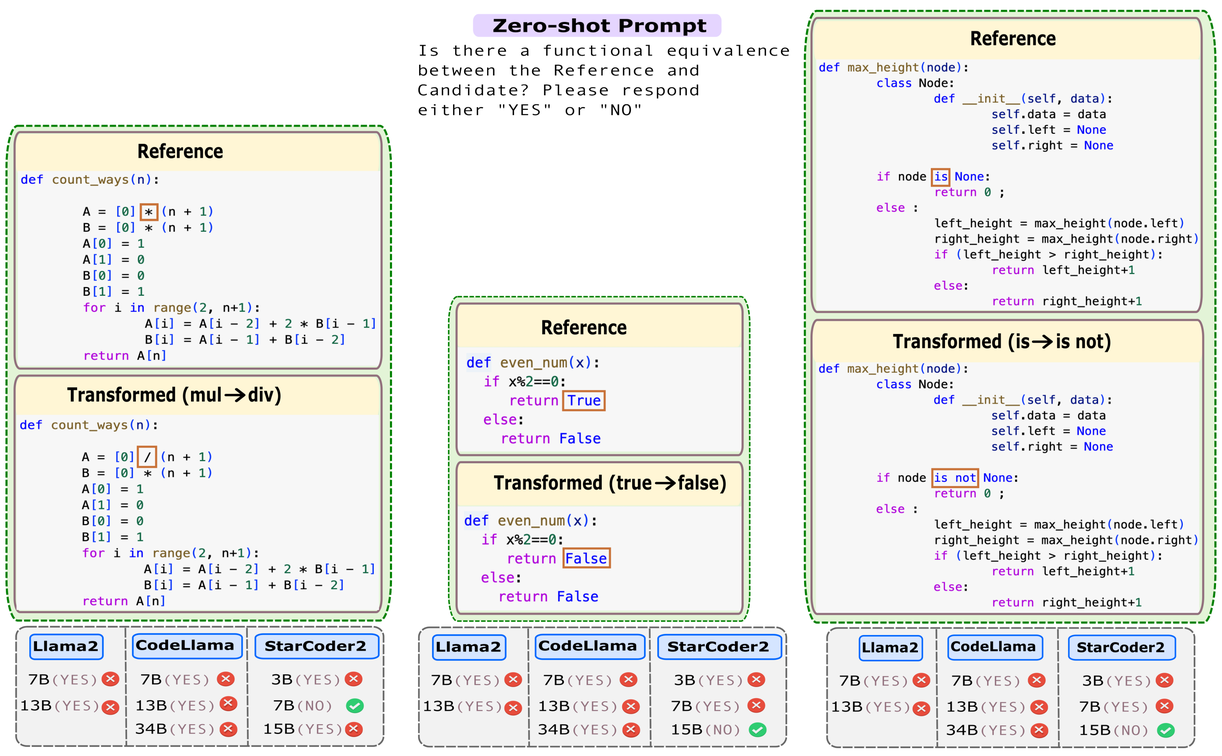
- It cannot scale to complex codebases that resemble real-world software domains.
- It is practically infeasible to cover all possible inputs, edge cases, and execution traces.
- Executing code, especially from an unknown source, can give rise to security risks.

Approach

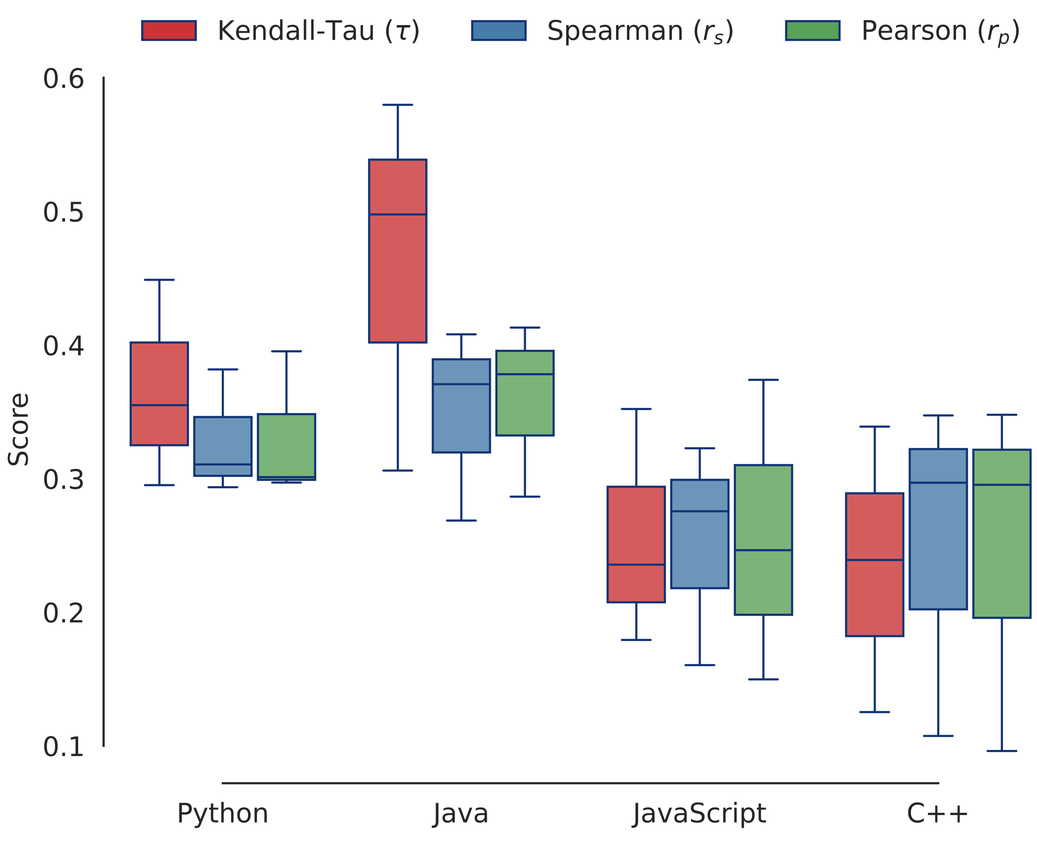
1. We propose **SeqCoBench**, a benchmark for assessing code evaluation metrics (CEMs) on the code functional equivalence task, constructed using the MBPP dataset.
2. We create over 20 code transformations to preserve or alter the code semantics.
3. We use match-based and LLM-based CEMs to produce a similarity score on the transformed code snippets.
4. We conduct extensive evaluations in different settings, including zero-shot (w/ prompting) and parameter-efficient fine-tuning methods.

CodeLLMs can be brittle

CodeLLMs do not fully capture the underlying semantics as they struggle to identify subtle changes involving arithmetic, boolean, and identity operator misuse.

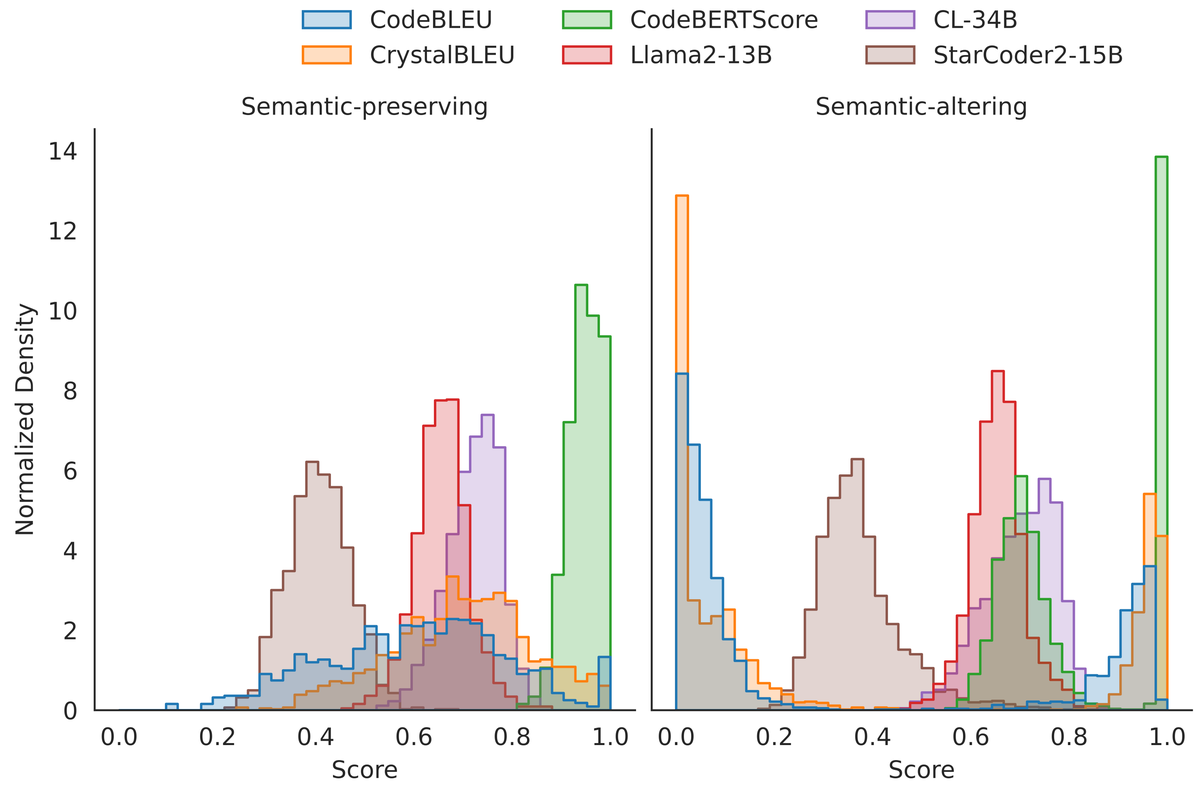


Correlation of LLM-based metrics with functional correctness: Current LLM-based metrics, like CodeBERTScore and CodeScore, show minimal correlation with functional correctness on HumanEval across multiple languages, with no metric exceeding an average correlation coefficient of $r = 0.31$, highlighting a significant opportunity for developing better CEMs.

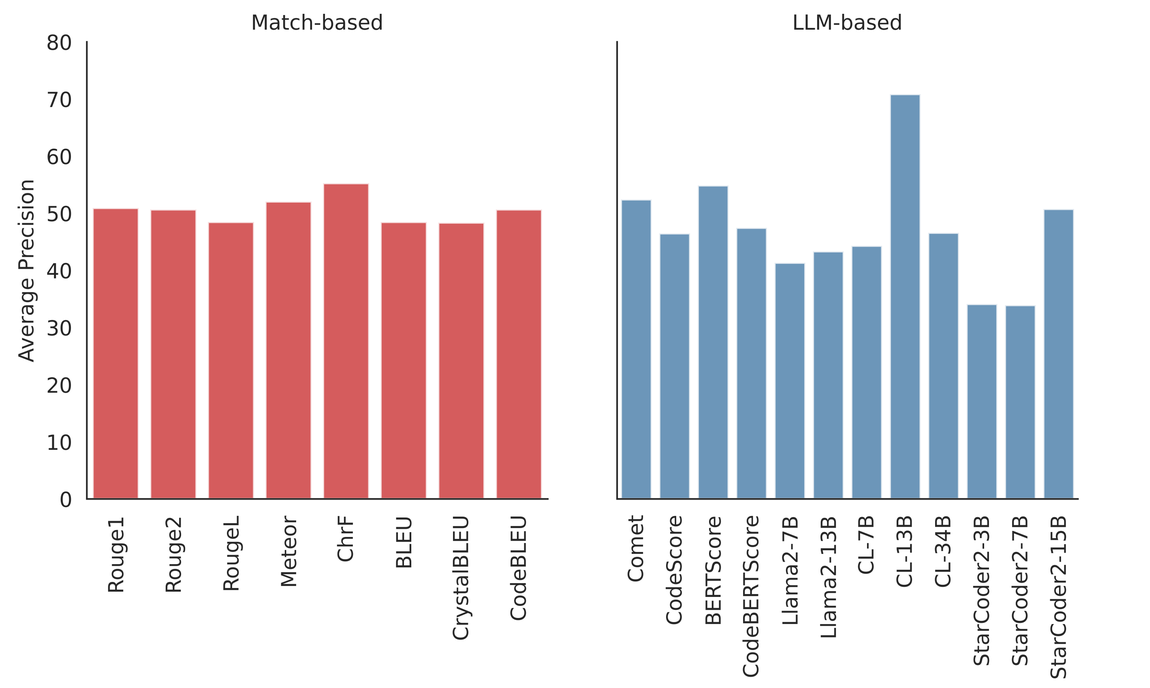


Result highlights

Normalized histogram statistics: Considering the metrics, we observe peaks in scores above 0.9 for semantic-altering (SA) transformations, leading to incorrect semantic similarity calculations, with the probability distribution of scores for SA labels often higher than for semantic-preserving (SP) labels.



Zero-shot evaluation using SeqCoBench: LLM-based metrics struggle to differentiate between semantically equivalent and non-equivalent code snippets, sometimes performing worse than surface-level match-based metrics, indicating a lack of understanding of code semantics and reasoning based on underlying logic.



Impact of code transformations on evaluation metrics: LLM-based metrics struggle to classify SA transformations due to their susceptibility to subtle input variations, and our findings show that LLM variants specifically trained for coding tasks outperform their more general-purpose counterparts.