Pretraining on Logic-Programming Corpora Enhances CodeT5 Robustness to Adversarial Perturbations

Assignee Research

June 9, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: How does pretraining on logic-programming corpora like Prolog affect the robustness of CodeT5 against unrestricted adversarial perturbations in code generation tasks compared to imperative language. 16 claims were extracted from source literature; 0 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 3.2/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Test Case Generation for Object-Oriented Imperative Languages in CLP. Research question: How does pretraining on logic-programming corpora like Prolog affect the robustness of CodeT5 against unrestricted adversarial perturbations in code generation tasks compared to imperative language corpora?.

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 3.2/10.

3 Results

14 papers retrieved. 16 claims extracted; 0 independently verified. Quality review score: 3.2/10.

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
The presented techniques are implemented and integrated in the PET tool.	×	0.03
The PET tool is available for download and online use at http://costa.ls.fi.upm.es/pet.	×	0.01
The first benchmark group comprises four classical programs taken from Charreteur and Gotlieb.	×	0.03
The second benchmark group consists of seven programs selected from the net.datastructures library.	×	0.02
The net.datastructures library is a library of algorithms and data-structures for Java.	×	0.04
Experimental times were obtained as the arithmetic mean of five runs.	×	0.00
Experiments were conducted on an Intel Core 2 Quad Q9300 at 2.5GHz with 1.95GB of RAM running Linux 2.6.26 (Debian lenny	×	0.03
Code coverage measures the percentage of bytecode instructions exercised by test cases among all reachable instructions.	×	0.05
The block-2 configuration achieves approximately 100% code coverage for the first 8 benchmarks.	×	0.03
For the class Sort, block-2 is not sufficient to reach some parts of the code.	×	0.02
The block-3 configuration achieves 100% code coverage for the class Sort.	×	0.03
Some methods reachable from NodeList and SortedListPriorityQ contain unreachable code (dead code).	×	0.02
For the Josephus benchmark, depth-50 generates 1 test case in 6 ms.	×	0.08
For the Josephus benchmark, depth-50 exercises only 56% of the code.	×	0.02
For the Josephus benchmark, depth-200 achieves 100% code coverage.	×	0.03
For the Josephus benchmark, depth-200 takes 366 ms.	×	0.00

References

- http://arxiv.org/abs/2303.12869v1
- http://arxiv.org/abs/1007.5195v1
- http://arxiv.org/abs/1701.08119v1