CodeT5 Pretraining on Procedural Code vs. Natural Language for Zero-Shot Security Vulnerability Detection

Assignee Research

June 7, 2026

Abstract

This report synthesises findings from 11 peer-reviewed papers addressing the following research question: What is the impact of pretraining CodeT5 on procedural code data versus natural language data on its zero-shot reasoning capabilities for security vulnerability detection in Python. 13 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: Case Study: Fine-tuning Small Language Models for Accurate and Private CWE Detection in Python Code. Research question: What is the impact of pretraining CodeT5 on procedural code data versus natural language data on its zero-shot reasoning capabilities for security vulnerability detection in Python?.

2 Methodology

Systematic literature search across multiple databases yielded 11 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

11 papers retrieved. 13 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
Farasat & Posegga [15] achieved an average accuracy of 98.6%, F1 score of 94.7%, precision of 96.2%, recall of 93.3%, an	×	0.09
Bagheri et al. [31] reported an F1 score of 99% using a hybrid ML model (Self-attention + CNN - Conformer).	×	0.03
Singh et al. [32] achieved an accuracy of 0.66, precision of 0.65, recall of 0.66, and F1 score of 0.64 for CWE predicti	×	0.04
Dozono et al. [6] reported Python F1 scores for various LLMs: GPT-4o=0.80, GPT-4T=0.76, Gemini 1.5 Pro=0.75, CodeLlama-7	×	0.01
Steenhoek et al. [34] evaluated LLMs on C/C++ and found a low balanced accuracy of 54.5%.	×	0.02
Shestov et al. [35] achieved an F1 score of 0.86 for binary classification using WizardCoder for JAVA CWE detection.	×	0.06
Li et al. [36] reported a 5-6% improvement over the base model using LoRa and IA3 fine-tuning approach for LLMs.	×	0.07
Jiang et al. [37] achieved the best F1 score of 87% using Llama 2-7b model with LoRa fine-tuning approach.	×	0.07
The un-tuned codegen-mono model failed to detect a single CWE within any of the code snippets presented in a baseline ev	×	0.12
The fine-tuned codegen-mono model achieved an accuracy of 99%, precision of $\approx$98.08%, recall of 100%, and F1-score of $\approx$99	✓	0.22
The methodology involves fine-tuning a Small Language Model (SLM) to specialize in detecting specific Common Weakness En	✓	0.20
The target vulnerabilities for the detection model were selected from the MITRE Top 25 Most Dangerous Software Weaknesse	×	0.06
The dataset was curated using a semi-supervised approach combining automated generation with human oversight, utilizing	×	0.06

References

- http://arxiv.org/abs/2504.16584v1
- http://arxiv.org/abs/2601.21725v2
- http://arxiv.org/abs/2603.02208v1