Sliding Window Truncation for Enhanced Defect Localization in LLaMA 3.2 on BugsInPy

Assignee Research

June 9, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: Does sliding window truncation preserve higher defect localization precision for LLaMA 3.2 on BugsInPy than fixed-head or fixed-tail truncation methods. 17 claims were extracted from source literature; 4 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 5.5/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: An Empirical Evaluation of Locally Deployed LLMs for Bug Detection in Python Code. Research question: Does sliding window truncation preserve higher defect localization precision for LLaMA 3.2 on BugsInPy than fixed-head or fixed-tail truncation methods?.

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 5.5/10.

3 Results

14 papers retrieved. 17 claims extracted; 4 independently verified. Quality review score: 5.5/10.

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
Locally executed models achieve accuracy between 43% and 45% in bug detection.	✓	0.27
Locally executed models produce a large proportion of partially correct responses that identify problematic code regions	✓	0.32
Performance varies significantly across projects, highlighting the importance of codebase characteristics.	✓	0.24
Local models can identify a meaningful share of bugs, though precise localization remains difficult for locally executed	✓	0.33
Performance degrades when bugs require cross-function reasoning.	×	0.05
Defect complexity is the primary factor governing detection accuracy.	×	0.03
Model performance drops systematically as the number of co-occurring defects increases.	×	0.03
Open-weight models can approach proprietary system performance on converting unstructured bug reports into structured fo	×	0.04
The availability of open-weight models such as LLaMA and Mistral has made local deployment a practical option.	×	0.07
Prior work has largely evaluated large cloud-hosted models.	×	0.07
Codex, a GPT-based model fine-tuned on publicly available code, evaluates its ability to generate functional Python prog	×	0.05
CodeBERT, trained jointly on natural language and programming language data, enables tasks such as code search and docum	×	0.06
AutoFL uses LLMs for fault localization while generating natural language explanations alongside predictions, improving	×	0.05
BugsInPy is a curated set of real Python bugs with reproducible test cases drawn from well-known open-source projects.	×	0.06
Reproducibility issues were identified in the BugsInPy dataset and revisions were proposed to support more reliable eval	×	0.03
Evaluation was extended to multi-vulnerability settings, showing that model performance drops systematically as the numb	×	0.03
A systematic review of LLM-based automated program repair was conducted, categorizing methods and identifying open chall	×	0.04

References

- http://arxiv.org/abs/2105.10886v1
- http://arxiv.org/abs/2310.00905v2
- http://arxiv.org/abs/2604.23361v1