Repository Context Integration in Multimodal GNN-Based Code Completion on BIGCode

Assignee Research

June 2, 2026

Abstract

This report synthesises findings from 14 peer-reviewed papers addressing the following research question: What is the impact of integrating repository context (e.g., imports, parent classes) on the accuracy of code completion tasks when using multimodal GNN-based models evaluated on the BIGCode benchmark. Retrieval-augmented generation (RAG) has recently demonstrated considerable potential for repository-level code completion, as it integrates cross-file knowledge with in-file preceding code to provide comprehensive contexts for generation. To better understand the contribution. 18 claims were extracted from source literature; 2 were independently verified against retrieved documents. An automated multi-reviewer quality assessment produced a score of 4.7/10. This report is a machine-generated literature synthesis and does not constitute original research.

1 Introduction

This paper examines: Impact-driven Context Filtering For Cross-file Code Completion. Research question: What is the impact of integrating repository context (e.g., imports, parent classes) on the accuracy of code completion tasks when using multimodal GNN-based models evaluated on the BIGCode benchmark?.

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 4.7/10.

3 Results

14 papers retrieved. 18 claims extracted; 2 independently verified. Quality review score: 4.7/10.

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 model is evaluated on two benchmarks: RepoEval and CrossCodeLongEval.	×	0.07
RepoEval includes line, API, and function completion tasks derived from 14 high-quality Python repositories.	×	0.07
CrossCodeLongEval extends the repositories from CrossCodeEval to include chunk-level and function-level code completion	×	0.13
Two completion settings are considered: Infilling and Left-to-right.	×	0.02
Infilling involves completing the middle part of the code based on both the preceding and subsequent context.	×	0.05
Left-to-right involves generating code sequentially using only the preceding context.	×	0.05
The evaluation uses the execution-based metric pass rate of Unit Tests (UT) for function-level completion in the RepoEva	×	0.08
For all other data, the evaluation uses the reference-based metrics Exact Match (EM) and Edit Similarity (ES).	×	0.04
CODEFILTER is compared against three baseline retrieval configurations: No Retrieve, Full Retrieve, and RepoFormer.	×	0.04
No Retrieve uses only in-file contexts for code completion.	✓	0.16
Full Retrieve augments in-file contexts with the top 10 cross-file code chunks.	✓	0.16
RepoFormer decides whether retrieval is needed before initiating it.	×	0.03
CODEFILTER consistently achieves notable enhancements across various tasks compared to full and adaptive retrieval metho	×	0.11
In the infilling setting, the performance of StarCoderBase-3B under CODEFILTER framework is comparable to, or even surpa	×	0.03
The gains achieved by CODEFILTER in the infilling setting mirror those observed when negative chunks are removed from th	×	0.06
The performance of StarCoderBase-3B with CODEFILTER is 65.19 in one setting.	×	0.02
The performance of StarCoderBase-7B with CODEFILTER is 51.11 in one setting.	×	0.02
The performance of CodeLlama-7B with Retrieve is mentioned in Table (p16).	×	0.01

References

- http://arxiv.org/abs/2406.00552v4
- http://arxiv.org/abs/2508.05970v1
- http://arxiv.org/abs/2310.11248v2