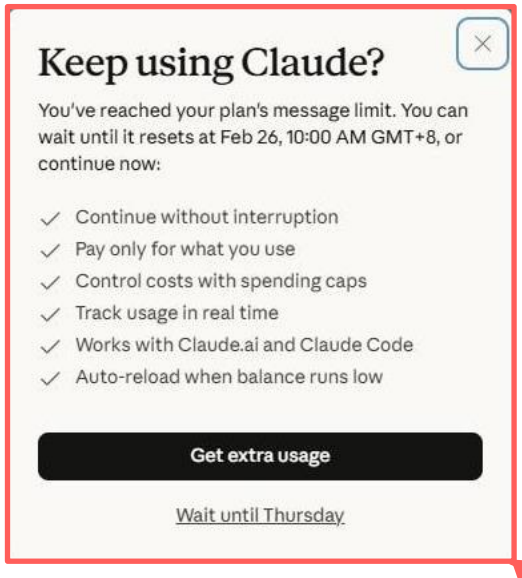
UNIVERSITY OF CALGARY

THE BUG-LOCALIZATION BOTTLENECK

If AI can't find the bug, it can't *fix* it.



–94%

Statement-level Top-5 accuracy when file-level localization is removed[1]

- Real-world software repositories are massive – thousands of functions, hundreds of thousands statements.
- Localization from bug reports is difficult.
- Unbounded traversal inflates tokens, latency, and dollars — without recall guarantees. It often can lead to this…

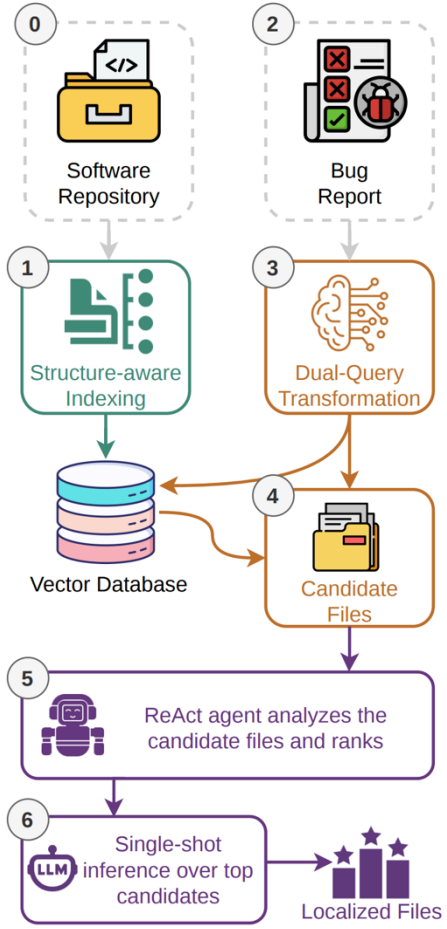
1. J. Chang, and et al. Bridging bug localization and issue fixing: A hierarchical localization framework leveraging large language models. IEEE Transactions on Software Engineering, 2026.

BLAgent – Agentic RAG for Bug Localization

by Md Afif Al Mamun and Dr. Gias Uddin

UNIVERSITY OF CALGARY

APPROACH



CONTRIBUTIONS

STRUCTURE-AWARE INDEXING

Abstract Syntax Tree-based chunks + file-path tags

File paths are prepended to every AST-aware chunk before embedding, so path-like tokens in bug reports — tracebacks, import errors, file references — directly align with their target chunk in the vector store.

DUAL-QUERY TRANSFORM

One bug report → Two complementary queries

Each bug report is decomposed into a structural query extracting code identifiers and module names, and a behavioral query capturing runtime symptoms and observed failures

TWO-PHASE AGENTIC RERANKING

Code skeleton scan → Evidence-anchored reranking

A ReAct agent scores candidates by inspecting file skeletons — class names, method signatures, docstrings — then a single LLM call over pruned implementation contexts confirms the final ranking

IMPACT ON LOCALIZATION

+20.5%

MRR improvement

over common text chunking techniques

Top-10 recall: 68.6% → 87.3%

94.3%

Top-10 candidate recall

compared to 87.3% using the base bug report

Structural: 93.0% · Behavioral: 91.0%

+15.9%

MRR from agentic reranking

two-phase reranking vs. basic LLM reranking

Basic RAG: 0.734 → BLAgent: 0.851

State-of-the-Art Accuracy — at a Fraction of the Cost

UNIVERSITY OF CALGARY

BENCHMARK

SWE-bench-Lite Dataset
300 Real GitHub Bugs
11 Popular Python Repos

TASK

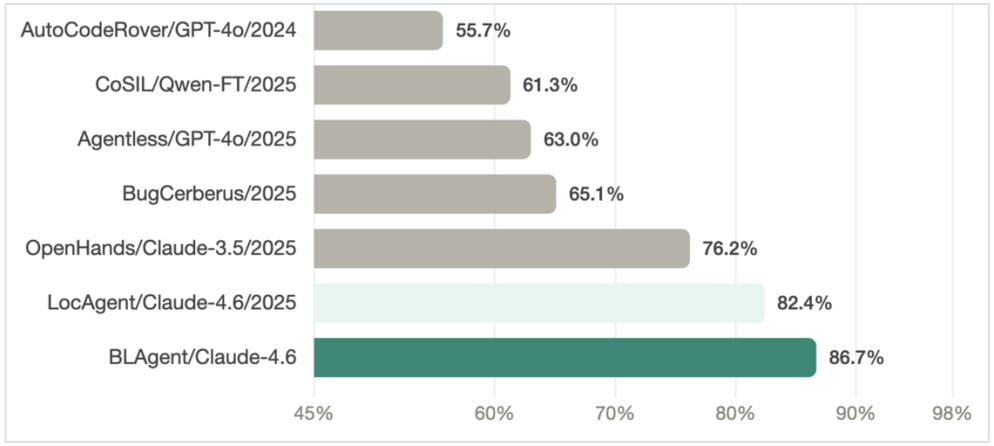
Identify the exact file that needs to be patched to fix a bug

METRIC

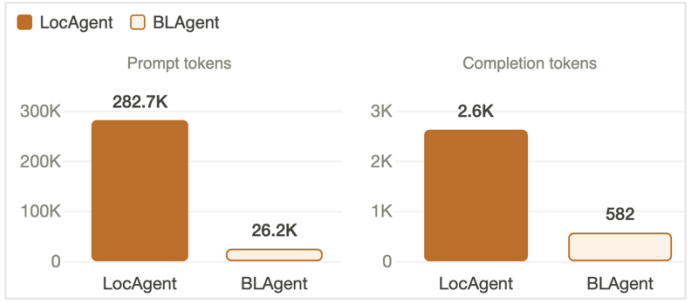
Top-1 File Localization Accuracy

No fine-tuning/training is required!

TOP-1 LOCALIZATION ACCURACY



TOKEN USAGE



COST & EFFICIENCY

18×

cheaper than LocAgent with Claude-4.6
$0.09 vs $1.65 per bug · $27 total for all 300 SWE-bench instances

Open-source models are competitive
Both outperform LocAgent (Claude 3.5) at 77.7%

OSS	Qwen3-32B	79.7%
OSS	GPT-OSS-120B	78.6%

< $1

to localize all 300 SWE-bench-Lite bugs
< $1 to localize all 300 bugs · $0.0017/bug with GPT-OSS-120B

Find Better — Fix Better

UNIVERSITY OF CALGARY

INTEGRATION

Agentless Automated Program Repair (APR) Framework (*OpenAI used it too*)

BENCHMARK

SWE-bench-Lite (300 bugs)

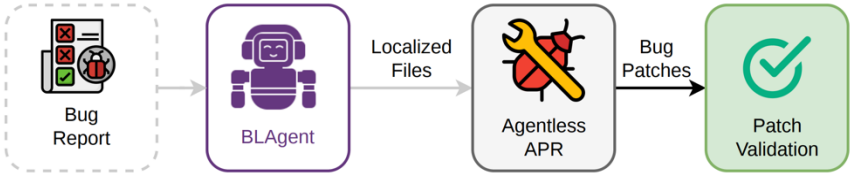
METRIC

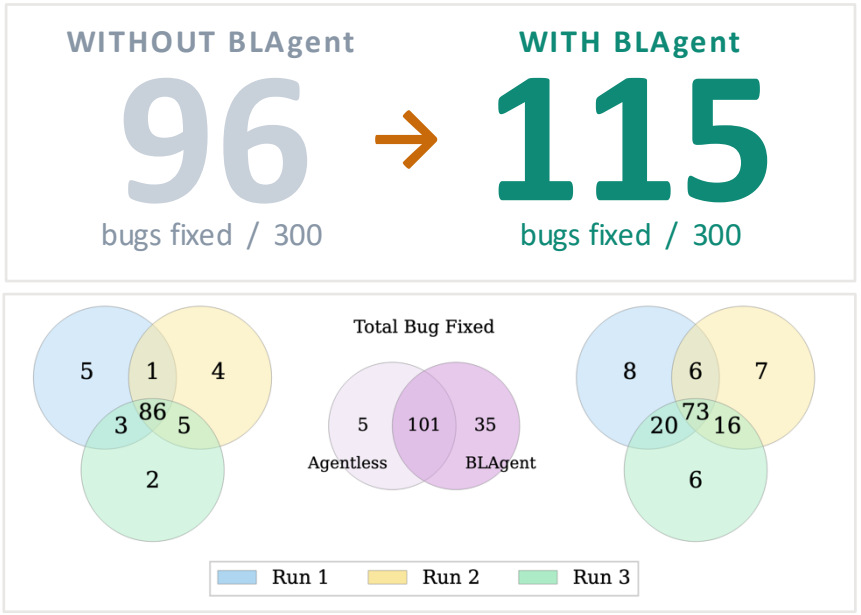
Repair Rate

LLM *for* Repair

GPT-OSS:120B

PROGRAM REPAIR EVALUATION





BLAgent helped fixing more bugs with the same APR framework and LLM across 3 independent runs.

RESEARCH OUTCOMES

+20%

end-to-end program repair rate

96 → 115 bugs fixed · Validated with Agentless on SWE-bench Lite

Plug-and-play with many APR system

Drop-in replacement for the localization step.
No retraining · Minimal integration overhead
Works with Agentless, SWE-agent, and more.

MCP-powered IDE integration

VSCode Copilot Inline localization as you code

Claude Code Agentic bug localization via MCP