CodeSense: Combining Static Analysis and LLMs for Code Review

Sowmya H.D
Assistant Professor
Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
sowmya.hd-cse@dsu.edu.in

S Rayyan Ahmed
Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
eng22cs0424@dsu.edu.in

Sahil K
Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
eng22cs0429@dsu.edu.in

Shreyas Bharadwaj
Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
eng22cs0459@dsu.edu.in

Suraj S
Computer Science and Engineering
Dayananda Sagar University
Bengaluru, India
eng22cs0425@dsu.edu.in

***Abstract*—Code quality is a decisive element of software maintainability and developer productivity. Conventional static analysis tools such as PyLint can detect syntax errors, code smells, and violations of coding conventions but tend to be lacking in contextual knowledge while proposing resolutions. To overcome this limitation, CodeSense has been designed, a hybrid code analysis tool that integrates static analysis with the reasoning abilities of a Large Language Model's (LLM). CodeSense is a Visual Studio Code extension that leverages PyLint for real-time linting and enhances the diagnostic output through passing the code and linter output to an LLM model. The model presents developers with a more cognitive and intuitive coding experience by generating contextual suggestions, quick fixes, and code improvement recommendations. An evaluation of the tool's performance was conducted on a number of Python projects, demonstrating improved coding efficiency, reduced manual debugging effort, and improved code quality. The system architecture, integration pipeline, and a qualitative performance analysis are discussed in this paper. Results indicate how deterministic and generative methods can be used together to create development environments that are smarter and more responsive.**

I. Introduction

The growing complexity of contemporary software systems requires strong tools to guarantee code quality, readability, and maintainability. Programmers must battle the challenge of producing functionally correct code while keeping to the best practices and reducing technical debt, especially during high-speed development cycles or pedagogy where experience and time are usually scarce. Static analysis tools like PyLint [6] are important by pointing out syntax errors, style faults, and possible bugs. Their rule-based approach normally resulting in lengthy diagnostics that need to be interpreted manually, and leading to unintended issues and decreased productivity.

Integrated development environments (IDEs) such as Visual Studio Code (VS Code) are at the core of modern workflows because they are extensible, allowing for smooth incorporation of linting, formatting, and version control tools. Notwithstanding these developments, solutions currently lack in offering real-time, context-aware guidance that closes the gap between static analysis and actionable code enhancements. Large language models (LLMs), such as CodeLlama [7], have been promising in code refactoring and code generation, as seen in tools such as GitHub Copilot [9]. Nevertheless, these tools are predominantly aimed at predictive code completion, with little integration with diagnostic feedback or structural code improvement support in the IDE.

To bridge these gaps, CodeSense in introduced, a VS Code extension that combines PyLint's static analysis with LLM-informed reasoning in order to provide smart, context-aware suggestions for code improvement. CodeSense activates PyLint on file saves to create diagnostics, which are then processed into structured prompts for a locally running CodeLlama model through Ollama [8]. The LLM produces accurate fix suggestions and refactorings presented directly in the VS Code editor.

This paper makes the following contributions:

- A hybrid VS Code extension combining static analysis with LLM-reasoning for improved code quality.
- A prompt engineering solution that makes use of PyLint diagnostics to inform LLM-produced fixes.
- A comparative study of CodeSense performance on benchmark Python datasets, measuring accuracy, latency, and developer productivity.
- Comparative study of LLM performance in producing actionable code fixes from static analysis results.

CodeSense is designed to automate the code review process, minimize manual debugging effort, and enable developers to write higher-quality code more quickly and effectively.

Applied for Conference Communication
Proof for acceptance
27/5/25


ASE 2025
Submission deadline in 46 hours · rayyanahmed2414@gmail.com
Search
(All)
Search
in Your submissions
Search
Advanced search
Display options
ID
Title
Status
#743 CodeSense: Combining Static Analysis and LLMs for Code Review
Submitted