ENHANCING STATIC CODE ANALYSIS WITH LARGE LANGUAGE MODELS

Oleksandr Huliakov, Maria Bilova
National Technical University «Kharkiv Polytechnic Institute», Kharkiv

Static code analysis plays a crucial role in software development by identifying potential issues without executing the code. However, traditional tools face challenges in balancing precision with comprehensive coverage and adapting to diverse coding patterns. Current solutions like ESLint and SonarQube, which are used for TypeScript and JavaScript, often rely on predefined rules and patterns that may miss context-specific issues or generate false positives when applied across different codebases.

Large Language Models (LLMs) have demonstrated impressive capabilities in various software engineering tasks including code generation and understanding. This work explores how LLMs can improve the process of static code analysis by offering deeper semantic understanding of code structures and behaviors. Unlike rule-based systems, LLMs can leverage their training on vast code repositories to recognize problematic patterns even when they don't match explicit rules, potentially identifying subtle bugs, security vulnerabilities, and maintainability issues that traditional analyzers might miss.

Proposed system implements a multi-stage analysis pipeline. The process begins with code processing, where source code is parsed and segmented into contextually coherent units that respect the token limitations of LLMs while preserving syntactic structures. Next, in the contextual enrichment phase, each code segment is augmented with project-specific context and customizable analysis rules. The prepared segments then undergo LLM-based analysis, during which the model identifies potential issues and generates improvement recommendations. Subsequently, through result aggregation, recommendations from all segments are collected and consolidated into a unified set. The verification and refinement stage involves a second LLM pass that validates and refines the recommendations, ensuring relevance, consistency, and clarity. Finally, output formalization structures and formats the results for best comprehension by final user.

By integrating the ChatGPT-o4-mini model, selected based on Chatbot Arena benchmarks for coding tasks, the system can interpret code context, apply project-specific standards, and provide meaningful recommendations. Also it allows to set up project rules using natural language and not specific semantics.

Preliminary evaluation through expert assessment shows promising results, with LLM-based analysis identifying issues missed by traditional tools while providing more contextually relevant explanations. This approach demonstrates potential for improving code quality while reducing the effort required to customize analysis for specific project needs.