MAJOR PROJECT

LITERATURE SURVEY

Automated Code Review using Machine Learning

In today's rapidly evolving software development landscape, our project introduces an "Automated Code Review System" that redefines and enhances the traditional code review process. This innovative tool leverages machine learning, static code analysis, and seamless integration into development pipelines to become a developer's trusted partner in pursuit of code excellence. Our primary aim is to provide developers with a comprehensive solution that not only assesses code quality but also detects vulnerabilities and enforces coding standards with precision. The system delivers real-time feedback and actionable suggestions, promoting ongoing enhancements in code quality. Its adaptability is a pivotal feature, allowing development teams to tailor review criteria to their unique coding practices. In an era of relentless development and continuous integration, our tool ensures immediate scrutiny of committed code, seamlessly integrating with version control systems like Git.

At its core, this tool merges static analysis and machine learning to recognize intricate bug and vulnerability patterns. Our project seeks to revolutionize the software development landscape by offering a robust solution that elevates code quality, enhances security, and streamlines the development process

Scope of the project:

Machine Learning in Code Analysis: To find out the ways of how machine learning techniques can be applied to code analysis, particularly in the context of identifying bugs, vulnerabilities, and code style violations.

Static Code Analysis: Explore the principles, methodologies, and tools employed for static code analysis, and assess their effectiveness in identifying code quality issues and vulnerabilities.

Integration into Development Pipelines: Research how code review systems are integrated into continuous integration and development pipelines. Analyze the benefits and challenges associated with real-time code review and its impact on software development processes.

Customization and Adaptability: Investigate how code review systems allow customization of review criteria to accommodate different coding practices and standards within development teams. Understand the importance of adaptability in addressing diverse coding norms.

User Interface and User Experience: Explore literature related to user interface design and user experience considerations in code review tools. Understand how intuitive interfaces can facilitate effective code review and foster developer engagement.

The specific research question or topic to be explored in our literature review could be formulated as follows:

"*How can machine learning and static code analysis techniques be effectively leveraged to enhance automated code review systems, with a focus on identifying bugs, vulnerabilities, and code style violations, and how can these systems be seamlessly integrated into development pipelines while allowing customization to accommodate diverse coding practices? ”*

Search Strategy:

Identify Relevant Databases and Libraries:

We begin by identifying key databases and digital libraries that are known for hosting relevant academic and industry-related sources in the fields of software development, machine learning, code analysis, and related topics. Some commonly used databases for such research include IEEE Xplore, ACM Digital Library, Google Scholar, PubMed (for security-related topics), and platforms like arXiv and ResearchGate. University libraries and institutional repositories may also provide access to valuable resources.

Select Appropriate Keywords and Search Terms:

We choose a set of relevant keywords and search terms that encompass the key aspects of our project. We consider using combinations of the following terms:

- Automated code review
- Code analysis
- Machine learning in code analysis
- Static code analysis
- Code quality assessment
- Code vulnerability detection
- Code style violations
- Integration into development pipelines
- Customizable code review systems
- User interface for code review tools

We can combine Keywords for more Comprehensive Searches like:

- "Automated code review" AND "machine learning"
- "Static code analysis" OR "code quality assessment"
- "Integration into development pipelines" AND "customizable code review"

Use Advanced Search Features:

We can make use of advanced search features provided by the selected databases to refine your search. These may include filters for publication dates, document types (e.g., research papers, conference proceedings), and specific fields (e.g., title, abstract).

Review Citations and References:

Once you've identified relevant sources, review the citations and references within those articles or papers. This can help you discover additional materials that may not have appeared in your initial searches.

<u>**Selection Criteria:**</u>

Relevance to the Research Question:

Sources should directly address or contribute significantly to your research question, which focuses on automated code review systems, machine learning in code analysis, static code analysis, code quality assessment, code vulnerability detection, code style violations, integration into development pipelines, customizable code review systems, and user interface design for code review tools.

Publication Date:

While older sources may provide foundational knowledge, prioritize recent publications (typically within the last five to ten years) to ensure that your review includes the most up-to-date information and reflects current trends and advancements in the field. Older sources may be included if they offer historical context or seminal contributions.

Credibility and Authoritativeness:

Evaluate the credibility and authority of the source. Academic journals, conferences, reputable publishers, and papers authored by experts in the field carry more weight. Peer-reviewed sources are preferred, as they undergo rigorous scrutiny by experts in the domain.

Methodology and Approach:

Consider the research methodology used in empirical studies. Assess whether the research methods are sound and appropriate for the research question. For instance, in studies related to machine learning, examine the quality of the datasets used and the rigor of the machine learning techniques employed.

<u>Data Extraction:</u>

Open-Source Code Repositories:

Platforms like GitHub, GitLab, and Bitbucket host vast amounts of open-source code projects. You can access code repositories on these platforms to gather a diverse range of code samples for analysis. Look for repositories related to programming languages, frameworks, or domains relevant to our project.

Code Review Tools:

Some code review tools and platforms, such as Phabricator, Gerrit, and Crucible, provide APIs or export options that allow you to access code review data. This data can include code changes, comments, and feedback from code reviewers.

Software Development Forums and Communities:

Online forums like Stack Overflow and specialized developer communities often contain code snippets and discussions related to coding issues, bugs, and best practices. These sources can provide valuable data for identifying common coding problems.

Bug Tracking Systems:

Bug tracking systems like Jira, Bugzilla, and GitHub Issues store information about reported bugs and issues. You can extract data from these systems to identify code issues and their resolutions.

Code Snippet Websites:

Websites like GitHub Gists or Pastebin host code snippets shared by developers. You can search for specific types of code issues or examples on these platforms.

Research Datasets:

Some academic researchers share datasets related to code quality, code reviews, and software vulnerabilities. These datasets can be valuable for training machine learning models.

Identification of gaps:

Effectiveness of Machine Learning Models: The literature lacks comprehensive studies comparing the effectiveness of various machine learning models in identifying specific code issues like bugs, vulnerabilities, and code style violations.

Integration into Real-Time Development Pipelines: While integration of code review into development pipelines is discussed, there's a gap in understanding the practical challenges and best practices for seamless integration, especially in large-scale continuous integration environments.

Security Vulnerability Detection: Existing research primarily focuses on code quality and style, leaving a gap in the specialized area of security vulnerability detection. Further research is needed to develop advanced techniques for identifying and mitigating security risks during automated code reviews.

Critical Evaluation:

Author's Qualifications:

Consider the qualifications and expertise of the authors. Are they recognized experts in the field of automated code review, machine learning, or software engineering?

Publication Venue:

Examine the publication venue (e.g., journals, conferences, academic publishers). Reputable venues have stringent peer-review processes that ensure the quality and validity of the research.

Research Methodology:

Evaluate the research methodology used in empirical studies. Assess whether the methods employed are rigorous and appropriate for the research question.

Relevance to Your Research Question:

Assess how closely the source aligns with your research question and objectives. A source's relevance is a key factor in determining its quality for your project.

Conclusion :

In conclusion, the literature review underscores the use of machine learning and static code analysis in Automated Code Review Systems, focusing on bug detection, vulnerability identification, and code style enforcement. While studies emphasize real-time integration into development pipelines, customization, and user-friendly interfaces, gaps exist in understanding the effectiveness of machine learning models, integration complexities, security vulnerability detection, and human-machine collaboration. Further research is imperative to address these gaps, facilitating the development of more effective code review tools that enhance code quality, security, and the software development process in a rapidly evolving landscape.