Project Title

"Automated Bug Detection and Fixing in C Programming using ML"

Department of Computer Science and Engineering,

Sri Sairam college of Engineering

Team members' names:

1. **G Chinthan**
2. **NityaShree**
3. **Jana Srihitha**

Abstract

This project presents an automated system for detecting and suggesting fixes for common bugs in C programming language code. The system combines static code analysis with machine learning techniques to identify potential issues such as memory leaks, buffer overflows, null pointer dereferences, and unsafe input/output operations. The solution includes a database-driven approach for storing code samples, detected bugs, and their corresponding fixes, making it a valuable tool for both educational and professional development environments.

Software bugs in C programs present critical challenges, leading to security vulnerabilities, performance degradation, and system failures. Traditional debugging techniques, including manual code reviews and static analysis, are time-consuming and often fail to detect deep logical errors. This project introduces an AI-driven Automated Bug Detection and Fixing System for C programming, leveraging deep learning models and generative AI to identify and resolve software bugs efficiently. The system integrates an AI-powered bug detection engine, an automated fix generation module, and seamless compatibility with development environments. By utilizing transformer-based models trained on extensive datasets, the solution enhances debugging accuracy and developer productivity. The expected benefits include reduced debugging time, improved software reliability, and scalability for broader programming language support. This project represents a cutting-edge approach to AI-assisted software debugging, streamlining development processes and ensuring high-quality code in C-based applications.

Introduction

C programming language, while powerful and widely used, is notorious for its susceptibility to various types of bugs and security vulnerabilities. Common issues include memory management problems, buffer overflows, and unsafe input/output operations. Manual code review and debugging can be time-consuming and error-prone. This project aims to automate the process of bug detection and provide intelligent suggestions for fixes, thereby improving code quality and security.

Problem Statement

The main challenges addressed by this system are: 1. Detection of common C programming bugs 2. Identification of security vulnerabilities 3. Generation of appropriate fixes for detected issues 4. Storage and management of code samples and their analysis results 5. Providing a user-friendly interface for code analysis

Software bugs in C programs pose significant challenges in software development, leading to system crashes, security vulnerabilities, performance degradation, and unpredictable behavior. Traditional debugging methods, such as manual code reviews, static analysis, and testing, require extensive time and effort, often failing to detect deep logical errors.

Furthermore, human error in the debugging process contributes to prolonged development cycles, increased maintenance costs, and potential failures in critical systems such as embedded devices, real-time applications, and security-sensitive software. The lack of an automated, intelligent approach to identifying and resolving software bugs hinders developers' efficiency and the overall software quality.

Recent advancements in Artificial Intelligence (AI) and Deep Learning (DL) provide an opportunity to enhance bug detection and automatic code fixing. By leveraging AI-driven techniques, developers can identify functional bugs, predict vulnerabilities, and generate reliable fixes in C programs. This project aims to develop an AI-powered system that automates the bug detection and fixing process, reducing debugging time and improving software reliability.

Project Description Department of Computer Science and Engineering,

Sri Sairam college of Engineering

The primary objective of this project is to design and implement an AI-driven **Automated Bug Detection and Fixing System** tailored for C programming. The system will utilize deep learning models and generative AI to analyze code, identify functional bugs, and provide recommended fixes to enhance software quality.

Key Features of the System:

✅ **Bug Detection Engine:**

- Uses AI models trained on vast datasets of C programs to identify syntax errors, logical bugs, memory leaks, buffer overflows, and undefined behaviors.
- Detects potential vulnerabilities and security loopholes such as null pointer dereferencing and uninitialized variables.

✅ **Automated Fix Generation:**

- Employs generative AI to suggest precise code corrections while maintaining program logic.
- Ensures that recommended fixes align with best coding practices and software security standards.

✅ **Seamless Integration:**

- Designed to integrate with popular **Integrated Development Environments (IDEs), CI/CD pipelines, and version control systems** for real-time bug detection.
- Enables developers to receive immediate feedback and suggestions for bug resolution.

✅ **Data Pipeline and Model Training:**

- Utilizes **large datasets of C code**, including open-source repositories and curated examples of buggy and fixed code.
- Implements **machine learning techniques** such as transformer-based models (CodeBERT, DeepBugs) to enhance detection accuracy.

✅ **Performance Evaluation:**

- Evaluates the system's accuracy using established metrics such as **Precision, Recall, F1-score**, and **Bug Fix Success Rate**.

- Continuously improves model performance using developer feedback and real-world testing scenarios.

Solution

The system implements a multi-component architecture: 1. **Code Analyzer**: A static analysis tool that uses regular expressions and pattern matching to identify potential bugs 2. **Database Management**: SQLite-based storage for code samples, bugs, and fixes 3. **Fix Generator**: Automated suggestion system for common bug patterns 4. **Main System**: Orchestrates the analysis and reporting process.

Key Features

- Pattern-based bug detection
- Severity classification of bugs
- Automated fix suggestions
- Database storage for analysis history
- Support for multiple bug types including:
    - Memory leaks
    - Buffer overflows
    - Null pointer dereferences
    - Uninitialized variables
    - Unsafe scanf/printf usage
    - Infinite loops

How It Works

- **Code Input**: The system accepts C source code files for analysis.
- **Static Analysis**: The code analyzer performs pattern matching using predefined regular expressions.
- **Bug Detection**: Identifies potential issues based on common bug patterns.
- **Fix Generation**: Suggests appropriate fixes for detected bugs.
- **Storage**: Stores analysis results in the database for future reference.
- **Reporting**: Generates detailed reports of detected bugs and suggested fixes.

Analysis Process

- File reading and preprocessing.
- Pattern matching for bug detection.
- Severity assessment.
- Fix suggestion generation.
- Database storage.
- Report generation.

Outputs of Solution

The system provides the following outputs: 1. **Bug Reports**: Detailed list of detected bugs including: - Bug type - Line number - Description - Severity level 2. **Fix Suggestions**: Automated recommendations for fixing detected issues 3. **Analysis History**: Database records of all analyzed code samples 4. **Severity Classification**: Categorization of bugs by impact level.

Demo of Execution

Here's a step-by-step demonstration of how the system executes:

- **Initialization**:

```
system = BugDetectionSystem()
```

- **File Analysis**:

```
bugs = system.analyze_file("example.c")
```

- **Output Example**:

```
Found 7 potential bugs:
- Potential null pointer dereference at line 8
- Memory leak at line 15
- Uninitialized variable at line 22
- Potential infinite loop at line 31
- Potential buffer overflow at line 38
- Unsafe scanf at line 45
- Format string vulnerability at line 55

Suggested fixes:
- For null_pointer at line 8:
  if (current != NULL) {
      while (current->next != NULL) {
          printf("%d ", current->data);
```

```
        current = current->next;
    }
  }
```
- For memory_leak at line 15:
```
  free(node);
```

Conclusion

This system provides an effective solution for automated bug detection and fix suggestion in C code. By combining static analysis with database management, it offers a comprehensive approach to improving code quality and security. The system can be particularly valuable for: - Educational institutions teaching C programming - Code review processes - Quality assurance testing - Security auditing - Professional development environments.

Future Enhancements

- Integration with more sophisticated static analysis tools.
- Machine learning-based bug pattern recognition.
- Support for additional programming languages.
- Enhanced fix suggestion system.

Real-time code analysis in IDE