MANTECH Publications
Recent Trends in Computer Science and Software Technology
Volume 5, Issue 3, September-December 2020

AI-Powered Code Generation and Debugging Tools

Dr. Kavita Sharma
Professor
Department of Computer Science
National Institute of Technology, Delhi, India
Email: *kavita.sharma@nitdelhi.ac.in*

Abstract

Artificial Intelligence (AI)-powered code generation and debugging tools have transformed the landscape of software development by automating routine programming tasks, improving accuracy, and accelerating project delivery timelines. These tools leverage advanced machine learning models, particularly large language models (LLMs), to generate high-quality source code, detect and fix bugs, and provide intelligent code suggestions in real-time. The adoption of AI in programming has enabled developers to focus on solving higher-order problems while minimizing time spent on repetitive coding activities. This paper explores the key technologies, applications, benefits, and challenges of AI-powered code generation and debugging tools, along with their potential to redefine the software engineering process.

Keywords*: AI, Code Generation, Debugging, Machine Learning, Software Development, Large Language Models, Automation*

INTRODUCTION

Artificial Intelligence has emerged as a disruptive force in software development, especially with the advent of AI-powered code generation and debugging tools. By integrating deep learning models trained on massive code repositories, these tools can produce syntactically and semantically correct code snippets, identify logical flaws, and suggest optimized solutions. Unlike traditional Integrated Development Environments

(IDEs) that provide static code analysis, AI-driven systems offer dynamic, context-aware assistance.

Technologies behind AI Code Generation

AI-powered code generation tools rely on transformer-based architectures such as OpenAI's Codex and Google's PaLM-Coder. These models are trained on billions of lines of code from open-source repositories, allowing them to learn syntax, semantics, and problem-solving strategies. Natural Language Processing (NLP) enables them to understand plain-English instructions, while reinforcement learning fine-tunes their output based on user feedback.

Applications in Software Development

These tools can assist in multiple stages of software development, including:

1. Automated code writing from natural language prompts.
2. Intelligent code completion and suggestions.
3. Real-time bug detection and automated debugging.
4. Code optimization for performance and scalability.
5. Automated documentation generation.

Advantages of AI-Powered Tools

AI-powered code generation and debugging tools offer:

- Increased productivity by reducing manual coding efforts.
- Enhanced accuracy and fewer bugs.
- Support for multiple programming languages.
- Real-time learning and adaptation to developer style.

Challenges and Limitations

Despite their advantages, these tools face limitations such as:

- Risk of generating insecure or inefficient code.
- Dependence on quality of training data.

- Potential intellectual property concerns.
- Need for human oversight to ensure correctness.

Future Directions

Future developments will likely focus on:

- Integration with DevOps pipelines.
- Stronger emphasis on security-aware code generation.
- Domain-specific AI models for specialized applications.
- Collaboration between AI and human developers for hybrid coding environments.

Table 1: Examples of AI-Powered Code Generation Tools

Tool Name	Developer	Key Features
GitHub Copilot	GitHub & OpenAI	Real-time code suggestions, multi-language support
TabNine	Codota	AI-based code completion using deep learning
Kite	Kite Inc.	Python-focused AI code completions

CONCLUSION

AI-powered code generation and debugging tools have the potential to transform software development into a highly efficient, collaborative, and innovative process. By automating repetitive coding tasks and providing intelligent debugging support, these tools enable developers to focus on complex problem-solving and creative aspects of programming. However, ethical and security concerns must be addressed to ensure responsible adoption. The future of programming will likely be characterized by a synergistic relationship between human creativity and AI capabilities.

REFERENCES

1. Vaswani, A., et al., 'Attention is All You Need', Advances in Neural Information Processing Systems, 2017.
2. Chen, M., et al., 'Evaluating Large Language Models Trained on Code', arXiv:2107.03374, 2021.
3. OpenAI, 'Introducing GitHub Copilot', 2021.
4. Google AI, 'Pathways Language Model (PaLM)', 2022.
5. Svyatkovskiy, A., et al., 'Intellicode Compose: Code Generation Using Transformer', 2020.
6. Li, Y., et al., 'Deep Learning for Code Completion', ACM Computing Surveys, 2022.
7. TabNine, 'AI Code Completion Overview', 2023.
8. Kite, 'Product Overview', 2021.