PharmaSUG China 2025 - Paper AI-135

AI-Assisted Programming: Theory, Practice, and Pathways to Enhanced Efficiency

Jian Qiao, Shenzhen Chipscreen Biosciences Co., Ltd.

ABSTRACT

AI-assisted programming significantly enhances code output efficiency and quality by leveraging artificial intelligence technologies. This paper explores the core concepts of AI-assisted programming, mainstream workflows (local deployment and cloud-based API calls), and proposes key principles for effective programmer-AI collaborative programming, including model selection, data security, budget planning, and precise instructions. Furthermore, the current study elaborates on a four-step AI-driven code generation process: requirement clarification, initial AI generation, manual review and feedback, and AI result revision. It aims to provide theoretical guidance and practical pathways for practitioners to fully utilize AI technology to empower programming work.

1.INTRODUCTION

With the rapid development of artificial intelligence technology, its application in various industries is deepening. In the field of statistical programming, AI-assisted programming has become an important means to improve productivity and optimize developgment processes. AI technology can assist developers in code auto-completion, intelligent error correction, and even the initial construction of complex logic, thereby significantly lowering the programming threshold, shortening development cycles, and achieving cost reduction and efficiency improvement. This manuscript aims to systematically review the basic concepts, key technical aspects, and practical strategies of AI-assisted programming to provide a reference for practitioners to efficiently utilize AI tools.

2.AI-ASSISTED PROGRAMMING: DEFINITION AND SIGNIFICANCE

AI-assisted programming, in essence, is the use of artificial intelligence technology to assist programmers in code writing and related activities. Its functions are extensive, ranging from basic code snippet auto-completion and intelligent detection and correction of syntax and logical errors, to more complex code generation, refactoring, and debugging support. For practitioners, the intervention of AI not only alleviates the burden of repetitive coding but also effectively reduces technical barriers and significantly shortens project delivery times by providing intelligent suggestions and automating parts of the process. Therefore, the value of AI-assisted programming in improving development efficiency and optimizing resource allocation is increasingly prominent.

3.MAINSTREAM AI PROGRAMMING WORKFLOWS AND THEIR COMPARATIVE ANALYSIS

Currently, the practice of AI-assisted programming mainly relies on two workflows:

3.1 LOCALIZED DEPLOYMENT OF AI MODELS

This model involves deploying and running AI models directly on the developer's local computer or private server.

- Advantages: The main advantage is a high degree of autonomy in data control. Since data does not leave the local

environment, it naturally offers advantages in terms of data security and privacy protection. At the same time, local deployment does not rely on external network connections and can operate stably in an offline environment.

- Disadvantages: It has high requirements for local hardware resources (such as CPU, GPU, and memory), and initial deployment and subsequent maintenance can be relatively complex.

3.2 CLOUD-BASED API INVOCATION

This model calls pre-trained and deployed AI models from cloud service providers through network interfaces.

- Advantages: It has extremely low requirements for users' local hardware configurations; almost any device with a network connection can use it. Cloud models are usually maintained by professional teams and can be updated rapidly, allowing users to continuously access the latest model capabilities and functional improvements.

- Disadvantages: It relies on the stability of the network connection. Data needs to be uploaded to the cloud for processing, which may raise concerns about data security and privacy, although service providers usually offer corresponding security measures.

The core goal of both workflows is to achieve efficient interaction between developers and AI models, so that the models can accurately understand requirements and generate code that meets expectations. The choice of workflow needs to comprehensively consider project requirements, data sensitivity, budget, and team technical capabilities.

4. THE CORE ESSENCE OF AI-ASSISTED PROGRAMMING: BUILDING EFFICIENT PROGRAMMER- AI COLLABORATION

To enable AI models to accurately produce the required code, building an efficient programmer-AI collaboration mechanism is crucial. The following principles are instructive for achieving this goal:

4.1 MODEL SELECTION

Different AI models have different focuses in their design and training, forming their respective areas of expertise. Although general-purpose AI models have certain code generation capabilities, their performance may not be as good as models optimized for code tasks when dealing with highly specialized or complex programming tasks. For example, some models are better at code understanding, generation, refactoring, and debugging. Therefore, selecting the most suitable model according to specific task requirements is the primary step to improve work efficiency. Given the rapid iteration of large language model technology, practitioners should stay informed about emerging models, continuously explore and evaluate their application potential in actual work, and keep pace with the times.

4.2 DATA SECURITY

The original intention of introducing AI-assisted programming is to improve efficiency, but if data security is sacrificed, the gain is not worth the loss. Data security must be prioritized. Before using any AI programming tools or services, it is essential to thoroughly understand the service provider's data processing policies, including whether data will be stored and whether it will be used for model retraining. To minimize potential risks, it is recommended to use desensitized or simulated data to interact with AI during development and testing phases, avoiding direct exposure of real core business data.

4.3 RATIONAL BUDGET PLANNING AND BENEFIT ASSESSMENT

Although there are free AI programming tools or API interfaces on the market that can handle simple tasks, they usually fall short in terms of model accuracy, processing power (such as input/output token limits), and service stability. For complex commercial applications or scenarios with high requirements for code quality, paid models often provide better performance, stronger contextual understanding, and more reliable technical support. Therefore, practitioners need to conduct a careful cost-benefit analysis based on project requirements, expected output quality, and available resources to determine an appropriate budget. It is particularly important to note that the limitation on the length of input text for models is a common bottleneck in current AI applications.

4.4 CLARIFICATION AND PRECISION IN INSTRUCTIONS

The quality of interaction with AI models directly affects the quality of their output. Vague, ambiguous, or incomplete instructions (i.e., prompts) often lead to the AI's inability to accurately understand user intent, thereby generating code that does not meet expectations. For the same programming problem, different wording and structures in the description can lead to vastly different solutions from AI. Learning how to construct structured, informative, and unambiguous instructions is a key aspect of improving the effectiveness of AI-assisted programming.

5. A STANDARDIZED FOUR-STEP PROCESS FOR AI-DRIVEN CODE GENERATION

In practice, programmers using AI for code generation can generally follow a structured four-step process to ensure collaboration efficiency and code quality:

5.1 STEP ONE: REQUIREMENT CLARIFICATION AND INPUT INFORMATION PREPARATION

The core of this stage is for the programmer to clearly and comprehensively convey task requirements to AI. Specific operations include:

- Target style definition: Clearly define the desired code output form or the final functional effect to be achieved (e.g., specific chart format).

- Data source and attribute specification: Detail the data source, structure, and key attributes that the code needs to process.

- Core variable identification: List the key variables in the code logic and their intended roles.

- Core logic elaboration (if necessary): For complex business logic, describe its calculation methods, judgment criteria, grouping basis, etc., in detail, such as the calculation method for key statistical indicators.

- Format specification setting: Clearly define the display precision of numerical values, date-time formats, and handling rules for special characters or outliers.

5.2 STEP TWO: AI EXECUTES INITIAL CODE GENERATION

Based on the detailed requirements and input information provided by the programmer, the AI model uses its learned knowledge and patterns to generate an initial version of the code. Some advanced models may also simultaneously provide simulations or descriptions of the expected results.

5.3 STEP THREE: MANUAL REVIEW, TESTING, AND FEEDBACK

This stage is a critical part of programmer-AI collaboration, where the programmer needs to rigorously review and test the AI-generated code. Key focus areas include:

- Functional correctness and completeness verification: Run the code to check if it can correctly achieve the intended functions and if the results are accurate.

- Code quality assessment: Check the readability, maintainability, and efficiency of the code, and whether it adheres to project coding standards.

- Format and style check: Confirm whether the output format and code style meet the preset requirements.

- Specific feedback: Avoid vague requests for modification; instead, point out specific problems, logical flaws, or parts of the code that do not meet expectations.

- Emphasize correct logic/format: Clearly reiterate the correct processing logic or desired output standard in the feedback.

- Decompose complex problems: If there are multiple or complex errors, it is advisable to break them down into several independent, specific modification requests so that AI could understand and correct them more accurately.

5.4 STEP FOUR: AI REVISES RESULTS BASED ON FEEDBACK

After receiving and parsing the programmer's feedback, the AI model makes corresponding adjustments and optimizations to the previously generated code. This "feedback-optimization" cycle may need to be repeated multiple times until the code quality and functional performance reach a standard satisfactory to the programmer.

This four-step process emphasizes close collaboration between programmers and machines. Through standardized interaction, it can effectively improve the efficiency and output quality of AI-assisted programming.

6. CONCLUSION AND OUTLOOK

AI-assisted programming, as a revolutionary technological advancement, is profoundly impacting data analysis work in clinical trials. Through diversified functions such as automated code generation and intelligent error detection and correction, AI significantly reduces the complexity of programming work and greatly improves development efficiency and code quality. To fully explore and utilize the great potential of AI-assisted programming, practitioners need to focus on precise model selection, strict protection of data security and privacy, rational planning of budget investment, and the ability to design instructions for efficient communication with AI.

The four-step method of programmer-AI collaboration proposed in this work—requirement clarification, AI initial creation,

manual feedback, and AI modification—provides a structured operational framework for practicing AI-assisted programming. By carefully designing input prompts, conducting meticulous code reviews and feedback, and combining this with a deep understanding of the AI model's capability boundaries, developers can effectively guide AI to produce high-quality code that meets complex practical needs.

Looking ahead, as model capabilities continue to enhance and related toolchains become increasingly sophisticated, the application of AI in the programming field will inevitably become more in-depth and widespread. From assisting in coding to automated testing, from intelligent project management to intelligent analysis and refactoring of codebases, AI technology is expected to play a greater role in various aspects of practical work.