Dean&Francis
ISSN 2959-6157

AI-Powered Coding Tools: A Study of Advancements, Challenges, and Future Directions

Dingyi Zhang

Abingdon School, Oxfordshire, United Kingdom

dingyi06.zhang@gmail.com

Abstract:

This review examines the effects of AI-assisted programming in contemporary software development, paying particular attention to tools driven by Large Language Models (LLMs), such as GPT-4o and GitHub Copilot. Starting with data processing and ending with model deployment, the review describes the standard workflow for training machine learning models and how programmers use these models to improve their coding processes. GitHub Copilot, an AI-powered code generator, and GPT-4o, a general-purpose LLM, are compared in terms of accuracy, usability, and efficiency when assisting with programming tasks. The results show that although both tools greatly facilitate coding, they each have particular advantages and disadvantages. GitHub Copilot is excellent at integrating with IDEs, providing contextual code recommendations and streamlining processes. In contrast, GPT-4o shows better accuracy when creating code from scratch, but it does not have Copilot's seamless IDE integration. The review also identifies some of the current drawbacks of AI-powered coding tools, including the possibility of producing faulty or vulnerable code as a result of training on unreliable datasets and the inability to comprehend context, which can occasionally result in functionally correct but practically incorrect code. In order to filter and fix problematic code before training, the review recommends using advanced algorithms for data pre-processing. It also suggests improving the interpretability of code generated by AI to help developers better comprehend and trust the results.

Keywords: Artificial intelligence; large language model; machine learning.

ISSN 2959-6157

1. Introduction

Programming has become an essential skill in modern human society, as it is being used in many different industries, including software, finance, healthcare etc. People's programming skill set are in high demand and are becoming more and more common in different careers. However, since the programming skill set itself requires a solid background in mathematics, familiarity in various algorithms and problem-solving techniques, the workforce of programmer cannot keep up with the growing demand of the current human resource market. The way that programmers manually code is also relatively inefficient as they will need to do repetitive work and might involve considerable time spent on debugging and refining code, which can slow down the working processes. In such a situation, a set of tools is required to assist the programmers' work, and this tool can be provided with the assistance of Artificial Intelligence (AI), especially Large Language Models (LLMs). These tools can assist programmers by providing a quick draft of the code to provide ideas, or to check existing code and provide suggestions related to improving the code.

One exact tool to help programmers code more efficiently and reduce error is Github Copilot, based on OpenAI's Deep Learning (DL) model Codex [1, 2], which is trained mostly over open-source Github code [3], is able to generate solution code and methods from existing code and code comments [4]. It has been shown to be useful for experienced programmers, by providing solutions for basic programming problems to reduce their workload [2]. However, it has demonstrated several drawbacks in various areas, including quality fluctuations when generating code in specific programming languages like JavaScript. Additionally, the code could be further simplified, often relies on undefined helper methods [5], and frequently produces inconsistent, buggy output [2], as well as vulnerable code [3].

Despite having the specialized tool Github Copilot, programmers use other general-purpose AI tools as well. An example of this is the most recent OpenAI LLM model, Generative Pre-trained Transformer 4 Omni (GPT-4o) released on 13 May 2024 [6]. Similar to Github Copilot, GPT-4o is able to generate computer code and refactor existing codes, with both tasks proven to be more success that previous models. Tests have been carried out to show these conclusions, but the code it manipulates has still been reported to fail when being applied [7]. Due to the importance of AI's assistance in both industrial region and educational region [1, 8], and the fact that more competitors such as Amazon CodeWhisperer [9], it is necessary for a review to be done in the region of AI-assisted programming to conclude the current situation, to help with future development of this region.

The remainder of this article consists as follows. This paper will firstly introduce the workflow of training a machine learning model, then provide an example of a typical programmer utilizing the tool of AI during the process of programming in Section 2. Followed by a discussion of the difference between different available AI-powered assisting tools, as well as some overall challenges faced by the entire AI-assisted programming region in Section 3. At the end, Section 4 will summarize the article, giving a general conclusion together with possible future directions of the region.

2. Method

2.1 Introduction of Machine Learning Model Training Workflow

In a standard machine learning model training workflow, the entire process begins with the collection of raw data and processing the data according to whether the learning method is supervised learning or unsupervised learning [10]. Then, the data is usually split into different datasets, such as training, validating and testing datasets. After preparing the data, the model of machine learning is set up by choosing a machine learning algorithm, defining the architecture of the model, and configuring hyperparameters. The next step is then to train the model with the previously prepared training dataset, and while the model is learning by adjusting its weights based on the input data and corresponding labels, the model's performance is evaluated by using the validating dataset to help adjust hyperparameters and prevent overfitting [10]. At the end, when the model achieves a certain level of performance, it will be tested using the testing dataset and if it passes the test with a certain score, it is then deployed for practical use, processing new data.

2.2 The Workflow of Programmer using AI in Programming

A typical process of a programmer utilizing AI tools begins with drafting code, where the programmer sets up an initial structure of the code, then an AI-powered code generator can produce foundational code snippets based on the programmer's initial input. The programmer will then either accept the generated snippet directly or take it as an inspiration to write their own code. As the programmer continues refining the code, an AI-driven code completion tool will anticipate possible next steps, offering context-aware suggestions that can speed up the coding process. At the same time, an AI-driven error-detection tool can recognize possible errors that the programmer has made, highlighting these erroneous codes to hint to the programmer for an immediate fix, and saving time for later debugging. Additionally, at the finishing phase

of writing a piece of code, LLM-based AI can assist the programmer by reviewing the code, providing feedback on the performance, readability, and compatibility of the code, and helping the programmer with any potential improvements.

2.3 Github Copilot

In the previously mentioned example, an AI-powered code generator and code completion tool are used to help the programmer complete the principal part of the code. Github Copilot is a popular example of such a tool. It is able to integrate into several Integrated Development Environment (IDE) and provide assisting functions for the programmer. One of the functions is the suggesting function, Github Copilot is able to give suggestions to either modify the current code or to add more code, determined by the existing context code and comments [11]. Another commonly used function is the IDE chat function, where Github Copilot gathers enter and context, and makes use of those factors to construct a prompt, which it then sends to the LLM, the LLM the generates a response which is returned to the programmer or Github Copilot. In the latter case, Github copilot will generate code according to the response from the LLM and return the code to the programmer. Common uses for this function are to let it explain a piece of code or to generate unit tests for the program [10].

2.4 GPT-4

Other than dedicated-purpose AI-powered tools, general-purpose AI-powered tools can also be used to assist programmers in coding, as one of OpenAI's newest LLM GPT-4o (based on the Transformer model) has been proven to be more effective on solving programming-related problems than previous existing LLMs [7]. Researches suggest GPT-4o and models before are being able to refactor existing code and significantly increase the quality of the code as well as the readability of the code, it is also suggested that GPT-4 is able to generate unit tests that has a great coverage [12]. Generally, GPT-4 and LLMs are able to improve code in general, explain the code, and eventually increase the efficiency of reviewing the code.

3. Discussion

3.1 Comparison between ChatGPT and Copilot

Analyzing the strengths and weaknesses of different AI-powered tools such as GitHub Copilot and GPT-4 in assisting programming involves assessing various key factors like code accuracy, user-friendliness and overall efficiency.

In terms of code accuracy, in a benchmark named HumanEval that aims to examine a model's ability to solve a python coding task, GPT-4 achieves a higher average score than Copilot, which suggests that GPT-4 is more capable of finish a single programming tasks accurately than Copilot [9]. However, the benchmark only considers the situation of basic programming tasks, not the complicated real-life programming tasks [13]. Different to GPT-4, copilot is more trained and designed towards generating code suggestions with context code, instead of generating code from scratch [11], and there is currently no benchmark or research that has been carried out to show which model is more accurate when generating code with context code. Therefore, though GPT-4 is generally more accurate in generating code than Copilot, it is still not clear which model is more accurate in actual application scenarios.

In terms of user-friendliness, since Copilot has a built-in support for multiple IDEs, it is significantly more convenient for the user to generate code directly inside their current working IDE than having to switch to a website to generate code from GPT-4 [11]. There are several third-party supports for GPT-4 inside several different IDEs, but the inconsistent update and the lack of official support will result in less friendly user interface and potential data security risks [14-16].

When it comes to overall efficiency, different AI-powered tools might perform differently under different tasks. Research has shown that, considering the code smell of the code these AI-powered tools generate, the average technical debt of Copilot is 9.1 minutes, and ChatGPT, slightly shorter, 8.9 minutes [9]. This has shown that the average working efficiency of both ChatGPT and Copilot is similar, but with ChatGPT being slightly more efficient.

3.2 Existing Shortcomings of AI-powered Coding Tools

Despite the recent advancements in AI-powered coding tools, several notable flaws still limit the effectiveness and reliability of these tools. The accuracy of the generated code is a related problem, as AI models such as ChatGPT is able to interpret and analyse existing code, and context given by the user, it is unable to develop the entirety of the context where the code is going to be employed. Therefore, it might not consider these contexts when generating codes, resulting in "functionally correct" but generally erroneous code [17].

The data that is used to train AI could also inflict issues in generated code. For example, Copilot has processed large amounts of unreviewed legacy code, which might lead to the existence of potential exploitable bugs in generated codes [3], or, if the training dataset of a model contains data that is a subject to copyright or other protections, codes that is involved in internal confidentiality of organizations or copyrighted information might be generated [17]. This dilemma is an important issue that needs to be resolved.

ISSN 2959-6157

3.3 Possible Future Prospects of AI-assisted Programming

AI-assisted programming has become an important domain in the context of advancing machine learning technologies. For the future of this field, this review suggests the following possible directions of development that AI-powered tools can improve on. Future AI models may not only use more sophisticated datasets but also incorporate advanced algorithms to pre-process and filter the training data to ensure better quality. For example, before feeding the data into the model, dedicated algorithms and AI-powered tools can be used to automatically filter and correct potentially faulty code, improving the integrity of the dataset, and increasing the degree of automation in the process of training and refining AI models. Additionally, improving the interpretability of AI models and the code they generated is crucial. By developing features that allow AI tools to generate algorithmic flowcharts or provide explanations for code suggestions, developers can better understand the reasoning behind the generated code. While these advances may increase the cost and complexity of training new AI models, they will significantly improve the quality of the code generated, as well as the usability of AI-based tools.

4. Conclusion

In this paper, the work of reviewing the role and currently existing flaws of AI in software development is finished. This paper introduced the workflow of both training and using programming-dedicated AI tools, as well as two different AI-powered tools, their advantages and disadvantages as an assisting tool. Currently, the domain of AI-assisted programming is facing the challenge of low-quality generated content. For possible future development routes of AI-powered tools, this paper suggests automating the process of pre-processing the training dataset as well as adding assistance features that increase the interpretability of these tools.

References

[1] Wermelinger M. Using github copilot to solve simple programming problems. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1; 2023 Mar; 172-178.

[2] Dakhel AM, Majdinasab V, Nikanjam A, Khomh F, Desmarais MC, Jiang ZMJ. Github copilot ai pair programmer: Asset or liability? J Syst Softw. 2023;203:111734.

[3] Pearce H, Ahmad B, Tan B, Dolan-Gavitt B, Karri R. Asleep at the keyboard? assessing the security of github copilot's code contributions. In: 2022 IEEE Symposium on Security and Privacy (SP); 2022 May; 754-768.

[4] Nguyen N, Nadi S. An empirical evaluation of GitHub copilot's code suggestions. In: Proceedings of the 19th International Conference on Mining Software Repositories; 2022 May; 1-5.

[5] Finnie-Ansley J, Denny P, Becker BA, Luxton-Reilly A, Prather J. The robots are coming: Exploring the implications of openai codex on introductory programming. In: Proceedings of the 24th Australasian Computing Education Conference; 2022 Feb; 10-19.

[6] Wiggers K. OpenAI debuts GPT-4o 'omni' model now powering ChatGPT. TechCrunch. 2024 Aug 5. Available from: https://techcrunch.com/2024/05/13/openais-newest-model-is-gpt-4o/

[7] Achiam J, Adler S, Agarwal S, Ahmad L, Akkaya I, Aleman FL, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774. 2023.

[8] Becker BA, Denny P, Finnie-Ansley J, Luxton-Reilly A, Prather J, Santos EA. Programming is hard-or at least it used to be: Educational opportunities and challenges of ai code generation. In: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1; 2023 Mar; 500-506.

[9] Yetiştiren B, Özsoy I, Ayerdem M, Tüzün E. Evaluating the code quality of ai-assisted code generation tools: An empirical study on github copilot, amazon codewhisperer, and chatgpt. arXiv preprint arXiv:2304.10778. 2023.

[10] Wang H, Lei Z, Zhang X, Zhou B, Peng J. Machine learning basics. Deep learning. 2016;98-164.

[11] GitHub copilot documentation. GitHub Docs. Available from: https://docs.github.com/en/copilot/, 2024.

[12] Poldrack RA, Lu T, Beguš G. AI-assisted coding: Experiments with GPT-4. arXiv preprint arXiv:2304.13187. 2023.

[13] Zhang S, Zhao H, Liu X, Zheng Q, Qi Z, Gu X, et al. NaturalCodeBench: Examining Coding Performance Mismatch on HumanEval and Natural User Prompts. arXiv preprint arXiv:2405.04520. 2024.

[14] Gitbito. Gitbito/Bitoai: Bito's AI helps developers dramatically accelerate their impact. it's a Swiss army knife of capabilities that can 10x your developer productivity and save you an hour a day, using the same models as chatgpt!. GitHub. Available from: https://github.com/gitbito/bitoai, 2024.

[15] Ismailkasan. Ismailkasan/chat-GPT-vscode-extension: CHATGPT assistant completion for vscode extension. GitHub. Available from: https://github.com/ismailkasan/chat-gpt-vscode-extension, 2024.

[16] Silasnevstad. Silasnevstad/GPT-extension-vscode: An extension bringing OpenAI's API to your fingertips inside of Visual Studio Code. GitHub. 2024. Available from: https://github.com/silasnevstad/GPT-Extension-VSCode, 2024.

[17] Atkinson CF. ChatGPT and computational-based research: benefits, drawbacks, and machine learning applications. Discover Artificial Intelligence. 2023;3(1):42.