WWW.IJRPR.COM

International Journal of Research Publication and Reviews

Journal homepage: www.ijrpr.com ISSN 2582-7421

Programmer's New Friend: GitHub Copilot

Aditya Babar

AISSMS Institute of Information Technology

A B S T R A C T

As AI will become extra popular in diverse industries including filmmaking (deepfakes), games, and many others, staring at how much AI affects coding became the principal motivation for this paper. GitHub Copilot is a newly emerging AI code completion software becoming very popular among programmers. Copilot uses natural language processing algorithms to generate the code based on the logic provided in the prompt or comments. According to its website, GitHub Copilot includes assistive features for programmers, such as the conversion of code comments to runnable code and autocomplete for chunks of code, repetitive sections of code, and entire methods and/or functions. It can generate boilerplate and repetitive code allowing the programmer to focus on more complex problems and logic in code. In this paper, we study the efficiency of the code generated by GitHub Copilot on its own and when it is working with a programmer and also study the code's readability. In terms of time complexity, the code generated by the copilot is 87.2% of the time as efficient as that written by a programmer, and 12.8% of the time it was able to generate more efficient code. The code generated by the copilot was much more like the code written by a human programmer in terms of code size, complexity, and Maintainability [2].

Copilot has about 91.5% success in generating a valid code. Code written with Copilot is comparable in complexity and readability to code written fully by human programmers.

Keywords: GitHub Copilot, Artificial Intelligence, Code Completion

1. INTRODUCTION

Today, programmers use an IDE (Integrated development environment) to put in writing code. this is because there are numerous utilities and equipment for writing applications efficaciously, which includes code completion, a feature in IDE that predicts what the programmer will type next.

Intelligent Code completion [4] is a context-aware code completion feature in a few programming environments that quickens the application coding system by reducing typos and other common errors.

these features use AI, ML, DL and neural networks for their implementation and one such software is his GitHub Copilot.

GitHub Copilot is an artificial intelligence device developed using GitHub and Open AI that supports customers of visible Studio Code, Visual Studio, Neo vim, and JetBrains incorporated development environments with code autocompletion [2].

1.1 Motivation:

As AI will become extra popular in diverse industries including filmmaking (deepfakes), games, and many others, staring at how much AI affects coding became the principal motivation for this paper.

There are a few AI-based code completion software programs like Tabnine and Kite, however, GitHub Copilot is a reasonably new and effective technology that has been demonstrated to be of first-rate

The benefit to programmers, so to see how much use it can be is necessary

1.2 Aim and Objective:

This task aims to combine data on Copilot's efficiency in writing code and the readability of the code it writes

Project objectives:

1. locating findings in code on GitHub Copilot performance

2. finding code clarity information on GitHub Copilots

3. Combining these insights into a single paper to reach the important conclusions

2. Literature Review

A few research empirically inspect different capabilities of Copilot. Yetiştiren et al. [1] examined the code generated by GitHub Copilot using a human eval dataset. They chose to go with paired programming approach and compared how well the GitHub copilot can act as a fellow programmer. They first let the copilot write the whole code on its own to get baseline results, then they first let the copilot act as a navigator during paired programming with the human programmer as the driver and then switched the roles. Sobania et al. [3] compared Copilot with genetic programming (GP)-based method that carried out an excellent performance in program synthesis. Their findings show that GP-based total procedures need extra time to generate a solution. moreover, training GP-primarily based fashions are pricey due to the high fee of records labeling. also, these techniques are not appropriate to help developers in practice as GP usually generates codes that can be bloated and difficult to recognize via human beings [3]. Vaithilingam et al. [4] carried out a human study involving 24 participants to understand how Copilot can help programmers to complete an assignment. They targeted 3 Python programming tasks: 1. edit CSV, 2. internet scraping" and 3. graph plotting". Those tasks contain much less problem-solving effort compared to the standard programming tasks in our study. they are in general associated with the use of programming language libraries. additionally, they did not evaluate the Copilot's recommendations with their members' guidelines when running without the assistance of the Copilot. Their finding shows that while Copilot did not always improve the assignment completion time and success rate, programmers prefer to use Copilot for their everyday tasks as it indicates the correct beginning factors to address the assignment Pearce et al. [5] carried out different situations in excessive-danger cybersecurity problems and investigated if Copilot learns from buggy code to generate insecure code. some other study investigates how Copilot can reproduce vulnerabilities in human programs [3]. To do so, they First used a dataset of vulnerabilities generated by humans, then rebuilt the whole code before the bug and requested Copilot to complete the code. The finished section was manually inspected with the aid of three coders to decide if Copilot reproduced the bug or fixed it. Naser et al. [2] In this paper, they consciousness on GitHub Copilot to deal with the problems of readability and visible inspection of model-generated code. readability and low complexity are crucial factors of proper source code, and visual inspection of generated code is vital when thinking about automation bias.

3. About GitHub Copilot:

GitHub Copilot as an “AI Pair Programmer” can generate code in different programming languages when provided with some context like comments, method name, surrounding code, etc.

According to its website, GitHub Copilot includes assistive features for programmers, such as the conversion of code comments to runnable code and autocomplete for chunks of code, repetitive sections of code, and entire methods and/or functions.

Its generated codes are 91.5% valid, and partially correct more than 50% of the time [1].

Code written with Copilot is comparable in complexity and readability to code written fully by human programmers [2].

It can generate boilerplate and repetitive code allowing the programmer to focus on more complex problems and logic in code.

Copilot can autosuggest solutions for the context you’re working in, but it can also understand natural human language in comments and function names, to synthesize a solution for your current task.

As the Copilot is trained on public repositories, if suggestions are based on popularity, then there is no guarantee that code suggestions are using the most up-to-date implementation, libraries, and approaches.

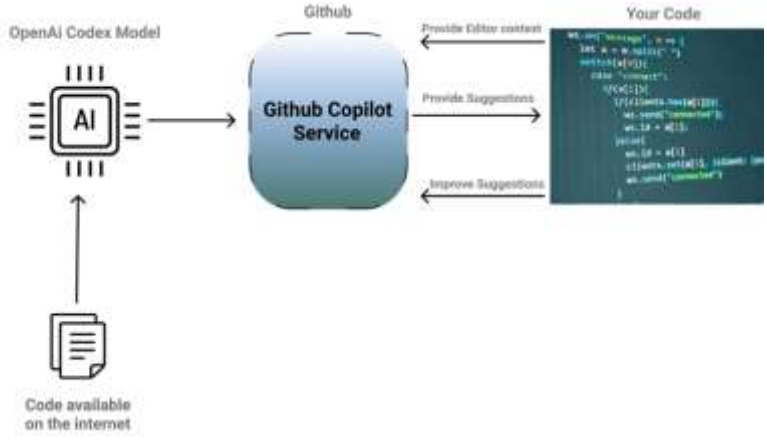

Fig 1: Working of GitHub Copilot

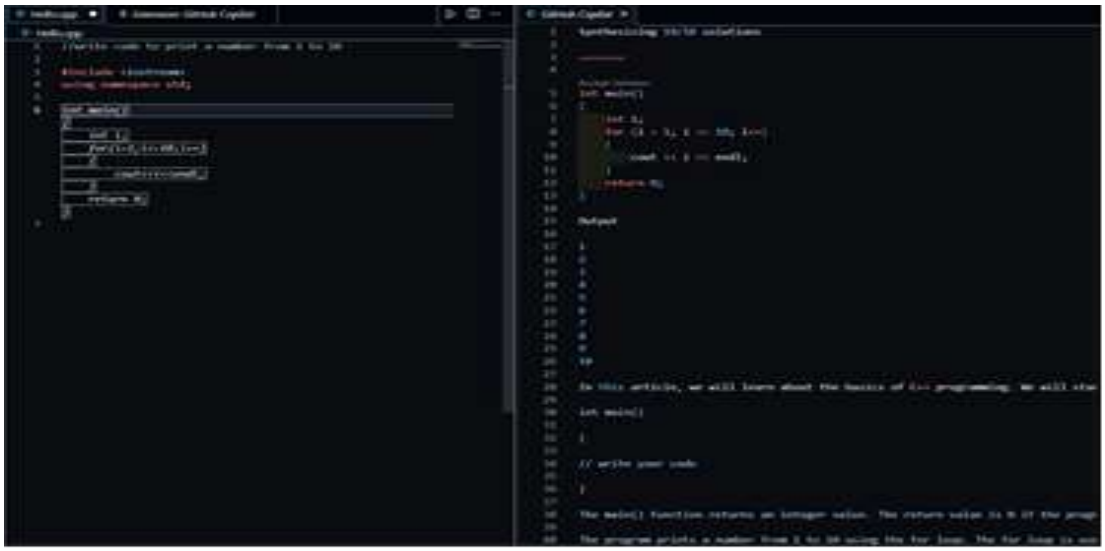

Fig 2: Using GitHub Copilot in VS code

4. Analytics:

- It has about a 91.5% success rate in generating a valid code [1].
- About 28.7% of the time, it generated the correct code, 20.1% of the time it failed to generate the correct code, and the remaining 51.2% of the time it generated partially correct code [1].
- In terms of time complexity, the code generated by the copilot is 87.2% of the time as efficient as that written by a programmer, and 12.8% of the time it was able to generate more efficient code [1].
- In terms of space complexity, for correctly generated code the efficiency is the same as provided in the canonical answer in about 89.7% of the problems, 4.2% of the results were more efficient and 6.4% less efficient than the canonical answer [1].
- The code generated by the copilot was much more like the code written by a human programmer in terms of code size, complexity, and Maintainability [2].
- As the AI is trained on the data on GitHub public repositories the code is much more readable and understandable.
- To generate more complex code there needs to be more depth in the prompt provided by the programmer
- Sometimes the code suggestion given by the Copilot is very lengthy and requires one to scroll up and down to view the entire code [2].

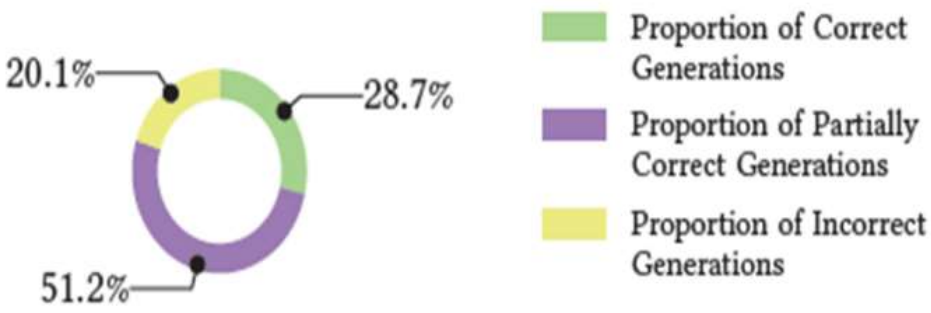

Fig 3.1: Distribution of code generation based on the correctness of code [1]

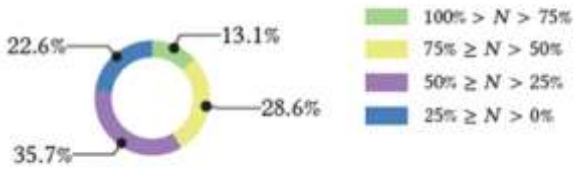

Fig 3.2: Distribution of Correctness Scores among Partially Correct Generations [1]

Table 1 - Efficiency Results in Time Complexity (TC) and Space Complexity (SC) [1]

	More Efficient		Same in Efficiency		Less Efficient		Total	
	TC	SC	TC	SC	TC	SC	TC	SC
Correct	6	2	41	42	0	3	47	47
Partially Correct	17	7	64	70	3	7	84	84
Valid incorrect	3	5	14	12	2	2	19	19
Invalid Incorrect	2	1	11	13	1	0	14	14

Table 1 - Comparing mean static code metrics of code written in each trial condition

	Copilot	**Driver**	**Navigator**
Code Size (LLOC)	19	11	15
Cyclomatic Complexity	3	2	2
Maintainability	23	24	27
Halstead vocabulary	13	9	12
Halstead volume	222	149	142
Halstead difficulty	3	1	2
Style error/LLOC	0.21	0.54	0.26

5. Conclusion

As seen from the above studies and analytical data provided by base papers, we can say that because of code's efficiency and readability generated by Copilot is very much like that of code written by a human programmer but the code's correctness is lower, and as it is trained on public data and we as a programmer don't exactly know what that data is there is always a possibility that the code suggested by Copilot is malicious, insecure or outdated. As the new version of Codex will be released it get even better in understanding natural language and generating correct code making it a very good and helpful AI Pair programmer.

Acknowledgments

We would like to express our special thanks of gratitude to our teachers, who gave us the opportunity to do this wonderful seminar on the topic of Programmer's New Friend: GitHub Copilot. It is our great fortune that we have got the opportunity to carry out this project work under the supervision of Dr. Meenakshi Thalor in the Department of Information Technology, AISSMS IOIT, affiliated with Savitribai Phule University. We express our sincere thanks and deepest sense of gratitude to our guide for his constant support, unparalleled guidance, and limitless encouragement. We would also like to convey our gratitude to all the faculty members and staff of the Department of Information Technology for their wholehearted cooperation to make this work turn into reality.

References

[1] Burak Yetiştiren, Işık Özsoy, Eray Tüzün, "Assessing the Quality of GitHub Copilot's Code Generation", 18th International Conference on Predictive Models and Data Analytics in Software Engineering (PROMISE '22), page no.4,11, November 2022

[2] Naser Al Madi, “How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot”, The 37th IEEE/ACM International Conference on Automated Software Engineering (ASE 2022) At Oakland Center, Michigan, United State, page no.4, October 2022

[3] D. Sobania, M. Briesch, and F. Rothlauf. Choose your programming copilot: A comparison of the program synthesis performance of GitHub copilot and genetic programming. arXiv preprint arXiv:2111.07875, 2021.

[4] P. Vaithilingam, T. Zhang, and E. L. Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In CHI Conference on Human Factors in Computing Systems Extended Abstracts, pages 1-7, 2022.

[5] H. Pearce, B. Ahmad, B. Tan, B. Dolan-Gavitt, and R. Karri. Asleep at the keyboard? assessing the security of GitHub copilot's code contributions. In 2022 2022 IEEE Symposium on Security and Privacy (SP) (SP), pages 980{994, Los Alamitos, CA, USA, May 2022. IEEE Computer Society. DOI: 10.1109/SP46214.2022.00057. URL https://doi.ieeecomputersociety.org/10.1109/SP46214.2022.00057