## Evaluation of Code Copilot's Quality and Maintainability in Code Explanation Generation and Impact Analysis

###### Research Background

In recent years, the development of machine learning and natural language processing technologies has provided programmers with powerful tools. OpenAI's Code Copilot is an AI-based code assistant that can generate useful code explanations based on context and code snippets. This study aims to explore the potential of Code Copilot in generating code explanations and further analyze the impact of these explanations on code comprehension and maintainability.

###### Research Objectives

The main objectives of this study are to evaluate the quality and practicality of code explanations generated by Code Copilot and investigate the influence of these explanations on regenerating code. We will compare the original code snippets, explanations generated by Code Copilot, and code snippets generated again by Code Copilot to assess the consistency, vulnerabilities, and comprehensibility between the explanations and regenerated code.

###### Research Methods

This study will be conducted in the following steps:

Data Collection: Select representative code snippets covering different programming languages and application scenarios as samples for the study. These code snippets will serve as the research samples. To facilitate analysis and comparison, code snippets can be categorized based on algorithm types or functionalities. Common algorithms such as sorting algorithms, search algorithms, graph algorithms, etc., and code snippets in different application scenarios like image processing, natural language processing, etc., can be chosen.

Explanation Generation: Use Code Copilot to generate explanations for the selected code snippets. These explanations will provide natural language descriptions of code functionality and implementation. When generating explanations, inputs provided to Code Copilot should include the code snippet itself and relevant contextual information, such as function or variable names, expected input-output formats, etc. This information can help Code Copilot better understand the meaning and purpose of the code.

Code Regeneration: Use Code Copilot to convert the generated explanations into code snippets. These regenerated code snippets will be compared and analyzed against the original code.

Vulnerability and Comprehensibility Analysis: Evaluate the differences in vulnerabilities and comprehensibility between the original code, generated explanations, and regenerated code. The analysis will include the detection of logical errors, potential vulnerabilities, and readability issues in the code.

Performance Evaluation: In addition to vulnerability and comprehensibility analysis, performance evaluation of the generated code can be conducted. This may include performance testing of the generated code, such as comparing execution time and resource consumption. This helps assess the efficiency and feasibility of the code generated by Code Copilot.

###### Research Plan

The anticipated research plan is as follows:

Weeks 1-2: Collect relevant literature and resources, gain a deep understanding of Code Copilot's working principles and existing research.

Week 3: Select sample code snippets, set up the experimental environment, and categorize the code snippets based on algorithm types or functionalities.

Week 4: Generate explanations using Code Copilot and record the results.

Week 5: Regenerate code using Code Copilot and compare it with the original code.

Week 6: Analyze vulnerabilities and comprehensibility of generated explanations and regenerated code.

Week 7: Perform performance evaluation of the generated code.

Week 8: Write the initial draft of the paper and conduct preliminary review.

Weeks 9-10: Revise the paper based on feedback from advisors and reviewers, and prepare the final draft.

Week 11: Submit the paper.

###### Expected Results and Contributions

Through this research, we expect to achieve the following results and contributions:

Evaluate the accuracy and practicality of code explanations generated by Code Copilot. We will analyze the consistency and richness of information between the generated explanations and the original code to determine if the explanations accurately describe the functionality and implementation of the code.

Analyze the differences and potential vulnerabilities between the regenerated code and the original code. We will compare the syntax and logic differences between the regenerated code and the original code, and detect potential vulnerabilities and errors.

Investigate the impact of generated explanations on code comprehension and maintainability. We will assess the extent to which the generated explanations aid in code comprehension, as well as their impact on code readability, maintainability, and extensibility.

Provide recommendations and guidance for the practical application of Code Copilot. Based on the research findings, we will offer best practices and considerations for utilizing Code Copilot to help developers make better use of this tool and improve code quality and maintainability in the software development process.

###### Reference

1. <https://docs.github.com/en/copilot/overview-of-github-copilot/about-github-copilot-for-individuals>
2. <https://dl.acm.org/doi/pdf/10.1145/3551349.3560438>
3. <https://dl.acm.org/doi/pdf/10.1145/3510454.3522684>
4. <https://dl.acm.org/doi/10.1145/3524842.3528470>
5. <https://dl.acm.org/doi/10.1145/3558489.3559072>

Problems:

1. copilot does not generate comments but only interprets the code
2. Most of the interpretations of the code found so far from github can be regenerated to produce the original code, I wonder if it has something to do with the fact that copilot has studied the code in github. Need some code snippets that are not publicly available on github.I hope the supervisor can provide some.