Investigating the Impact of Code Generation Tools (ChatGPT & Github CoPilot) on Programming Education

Faisal Nizamudeen, Lorenzo Gatti, Nacir Bouali and Faizan Ahmed
Faculty of Electrical Engineering, Mathematics and Computer Science, University of Twente, Enschede, The Netherlands

Keywords: ChatGPT, Artificial Intelligence, Code Generation, Computer Science Education.

Abstract: In our rapidly evolving technological landscape, AI tools have gained substantial power and integration across various domains. Through interviews and surveys conducted at a University in the Netherlands, we investigated students' perceptions of AI tools. Our results show that students generally have a positive attitude towards the adoption of AI technologies and feel that it enhances their learning experience. Furthermore, this research project examines the capabilities of AI-powered tools, namely GitHub Copilot and ChatGPT, in solving a variety of university-level assignments. By empirically evaluating the capabilities of these AI tools and offering insights to educators, this research project aims to assist them in designing programming exercises that encompass essential learning processes while accounting for students' utilization of AI tools. The findings indicate that a majority of the exercises currently utilized by the examined university could be solved partially or entirely with the aid of these tools. This project highlights the importance of educators understanding the capabilities of AI tools, as well as students' attitudes towards them, to effectively adapt their teaching methods and promote essential learning goals.

1 INTRODUCTION

The past year has seen a significant surge in AI innovation, with AI tools becoming increasingly integrated into people's daily lives worldwide. This trend is especially noticeable in the field of programming.

Code generation tools are models that can automatically generate source code or actual pieces of code based on input specifications, templates, or models (Herrington, 2003). They are accessible via plugins in integrated development environments (IDEs) and free-access websites that provide interaction with such generators. With their dramatic improvement over the past year, it is believed that they have the potential to revolutionize the coding process and make programming more productive and accessible (Li et al., 2022).

AI Chatbots and AI Pair Programmers are the most commonly used tools for code generation. Among these tools, ChatGPT (an AI Chatbot) and GitHub Copilot (an AI Pair Programmer) stand out as widely used applications. These technologies have notably diminished the reliance on platforms like Stack Overflow as these tools can offer personalized responses and better contextual understanding. As AI tools continue to advance, the importance of such sites will likely continue to diminish.

ChatGPT, created by OpenAI, is an AI chatbot that uses OpenAI's GPT model to simulate human-like responses when interacting with users. The model was trained using Reinforcement Learning from Human Feedback (RLHF), and a reward model was established based on conversations between trainers and the chatbot. While the potential applications of ChatGPT are virtually limitless, its implications for programmers are particularly significant. Within the programming context, programmers can rely on ChatGPT to generate code snippets, boilerplate code, stub implementations, and much more. It can also assist with debugging by providing relevant prompts and code snippets.

GitHub Copilot, developed by GitHub and OpenAI, is an AI-powered code completion tool that suggests code snippets and auto-completes code while developers write. It utilizes OpenAI's GPT (Generative Pre-trained Transformer) model to analyze the code being written, along with contextual information such as function names, comments, and documentation, to provide helpful suggestions. The tool aims to enhance coding speed and efficiency by reducing the need for manual typing. It has been trained on a vast amount of code from public GitHub repositories.

Nizamudeen, F., Gatti, L., Bouali, N. and Ahmed, F.
Investigating the Impact of Code Generation Tools (ChatGPT & Github CoPilot) on Programming Education.
DOI: 10.5220/0012628000003693
Paper published under CC license (CC BY-NC-ND 4.0)
In *Proceedings of the 16th International Conference on Computer Supported Education (CSEDU 2024) - Volume 2*, pages 221-229
ISBN: 978-989-758-697-2; ISSN: 2184-5026
Proceedings Copyright © 2024 by SCITEPRESS – Science and Technology Publications, Lda.

Students who apply for a GitHub Education Pack can access GitHub Copilot for free.

ChatGPT and Github Copilot are the primary technologies that we will focus on throughout this paper. We will delve into the impacts that these code-generation tools can have on education as a whole from the perspective of a student and educators. We analyze this impact by answering two main questions a) What do novice programmers say about the influence of code generation tools on their engagement and motivation in the programming learning process? b) To what extent can AI Tools assist students when completing programming assignments and how can teachers design exercises that still maintain relevant learning objectives?

The paper is organized as follows: Section 2 will provide the necessary background for this research. Section 3 describes the methodology adopted to conduct this research. In Section 4 we summarise our surveys and interviews. Section 5 is dedicated to exploring the capabilities of these tools. In Section 6 we provide the results of our study. We conclude this paper in Sections 7 & 8 touching upon the limitations of this study and providing concluding thoughts.

2 EXISTING WORK

In this section, we will briefly describe some of the existing work on the domain.

2.1 Impact on Educators/Students

Several studies have emphasized the potential of AI-generated code technologies to support teachers in automating exercise generation and providing explanations of the code.

(Sarsa et al., 2022) and (Marwan et al., 2019) have specifically highlighted the benefits of these tools in assisting educators in creating a wide range of programming exercises and offering detailed explanations, thereby enhancing student learning experiences. Similarly (Becker et al., 2022) strengthens the idea of using these tools as an aid for effective teaching. They also raise concerns about the readiness of teachers to handle the significant influence on educational practices. The researchers therefore emphasize the need for an urgent review and modification of instructional approaches and traditional practices in response to the advancements in code generation technology.

From the perspective of students or novice programmers, code-generating tools can aid in solving programming tasks (Wermelinger, 2023), resolving bugs (Surameery and Shakor, 2023), assisting with software engineering projects (Khmelevsky et al., 2012), generating exercises with explanations and illustrative examples of programming constructs and algorithmic patterns, or mapping problems to solutions (Becker et al., 2022).

Some studies identify the benefits of using code-generating technologies for students as improved performance and learning. The works of (Marwan et al., 2019) and (Ouyang et al., 2022) report increased productivity, consistency in code quality, and reduced errors when using those tools. A recent review of empirical studies reports an improvement in academic performance, online engagement, and participation when introducing AI applications in the educational process (Ouyang et al., 2022). Additionally, these tools can assist students in generating exemplar solutions for programming exercises and code reviews of solutions. Researchers point out the variety of solutions these tools can propose, allowing educators to introduce students to the diversity of ways that a problem can be solved.

(Dwivedi et al., 2023) in their paper delved into how some choose to label these AI systems as 'high-tech plagiarism' tools while also acknowledging the difficulty in detecting their use. Despite this, they believe that we should embrace the use of these systems as they have the potential to significantly benefit and enhance the learning experiences of the students.

On the other hand, identified drawbacks of code generation tools usage include the possibility of students having a lack of understanding of fundamental programming concepts. Several studies discuss the issue of bias, bad habits, and over-reliance. (Chen et al., 2021) and (Becker et al., 2022) raise concerns about its suitability for novices, given that public code is often contributed by professionals and may not reflect the expected quality for beginners and the desires of educators.

2.2 Capabilities of AI Tools

(Nguyen and Nadi, 2022) have evaluated the influence of programming languages on the accuracy of GitHub Copilot's code suggestions. For instance, in their study using GitHub Copilot on LeetCode problems, they found that the tool correctly solved 57% of the problems in Java, while only managing to solve 27% of the problems in JavaScript.

(Yetistiren et al., 2022) carried out an interesting study that revealed the significant impact of input parameters on the validity of the code generated by GitHub Copilot. Their findings indicated that the validity of the generated code dropped from 79% to 27%

when using dummy or sub-optimal functions and parameter names. (Dakhel et al., 2022) shared a similar viewpoint, concluding that the quality of the generated code depends greatly on the clarity and depth of the provided prompt. They also argued that GitHub Copilot can be a valuable tool for expert programmers but may present challenges for junior programmers who are still learning coding conventions and a programming language.

(Al Madi, 2022) also conducted research that focused on metrics related to the code generated by GitHub Copilot. The results indicated that the complexity and readability of code generated by GitHub Copilot were comparable to code written entirely by a human programmer.

(Wermelinger, 2023) argues that using GitHub Copilot to solve programming exercises can be a frustrating trial-and-error process. They suggested that while GitHub Copilot can provide an initial helpful attempt to solve a problem, students still require a solid understanding of a language's semantics to modify GitHub Copilot's sometimes incorrect suggestions.

On the other hand, ChatGPT offers a distinct approach to assisting students. ChatGPT can assist with a variety of tasks related to programming such as debugging, code optimization, code completion, error fixing, prediction, document generation, chatbot development, text-to-code generation, and technical query answering, as mentioned by (Biswas, 2023).

3 METHODOLOGY

The first research question is to be answered via the use of surveys and interviews with first-year Computer Science students. Interviews and surveys were selected as the investigation methods, aiming to gather firsthand experiences of novice programmers using code generation tools during their studies.

We received 39 survey responses and randomly chose 5 other first-year students as the interviewees. The students were randomly chosen and no prior checks were done before selecting candidates, for example, verifying their familiarity with these tools. The questions asked will query the students on their opinions and usage of these tools over the entire academic year. The interviews were conducted individually and in a closed setting.

Quantitative data from the survey was summarized using descriptive statistical analysis. Qualitative data from the interviews was analyzed using thematic analysis to identify common themes and patterns. Finally, we compared the gathered information to address the main research question, including a comparison of research findings from the literature overview and the outcomes of the conducted study.

The second research question was answered by carrying out experiments over a set of assignments currently used by a technical university in the Netherlands. Experiments were conducted using the latest version of GitHub Copilot and the regular (free) version of ChatGPT, which utilizes the GPT 3.5 model.

A diverse set of programming assignments was identified to ensure representative results. These assignments were selected from various first and second-year modules and encompass various paradigms of coding such as imperative, functional, and declarative styles.

Depending on the availability and ease of providing context, either ChatGPT or Copilot was utilized to generate code. On occasion, we would use both software to solve different parts of a large exercise. If the generated code contained errors or the program did not function as expected, ChatGPT was queried to detect and resolve the issues.

4 STUDENT PERSPECTIVE ON TOOL USAGE

In this section, we discuss how first-year Computer Science students are using code-generation tools and their subjective experience of using them. This was investigated with both a survey and an interview[1], with the goal of having a good understanding of the viewpoint of students. We focus in particular on the positive effects of these tools, and on the students' awareness of their risks in an educational setting.

4.1 Survey

4.1.1 General Familiarity

All 39 respondents demonstrated familiarity with ChatGPT, reflecting its recent popularity. ChatGPT was reported as the most used tool, with 77% mentioning practical usage.

In the realm of code generation tools, GitHub Copilot (69.2%), OpenAI Codex (30.8%), and Tabnine (15.4%) were widely known. DeepMind AlphaCode (7.7%), Microsoft CodeBERT, and Google PaLM (5.1% each) were less recognized and not used by survey participants. Notably, 15.4% had prior experience with code generation tools before starting

[1]Due to space constraints, the list of survey and interview questions is not included in this paper, but is available at https://drive.google.com/file/d/1nu6vdSpnKnwdRhVaI-3Z1YySXIoeefH7/view?usp=sharing.

their university Computer Science course, and half of them found ChatGPT most useful.

Only 6 out of 39 people (15.4%) tried code generation tools before starting their university computer science course. Among these individuals, two were quite familiar with code generation tools, and only one always used them in coding practices. Half of the students with prior experience in such tools marked ChatGPT as the most useful one.

4.1.2 First Year Computer Science Course

Out of all 39 responses, 21 (53.8%) have used code generation tools during their first year of the Computer Science course. Among these respondents, ChatGPT (GPT-4) was the most popular, with all respondents using it during their first academic year. GitHub Copilot is second on the list, used by 7 people (33.3%), followed by Tabnine (23.8%).

Concerning code generation tools specifically used within students' coursework, ChatGPT still leads with 66.6% of students utilizing it for study-related tasks. The second most popular response is using none of the tools (28.6%). Copilot takes the third place with (23.8%), while Codex and Tabnine each received only 1 vote (4.8%). Among respondents, 9.5% of people have used code generation tools frequently, and 61.9% used them at least a few times.

The next survey question revealed the specific programming tasks students used these tools for. The majority of votes went to project work and exam preparation (57.1% each), followed by assignments (47.6%), labs (33.3%), and group work (23.8%). Three people (14.2% of all respondents) reported not using any tool for any type of coursework.

A majority of students (76.2%) who completed the survey said that the use of code generation tools in their practices makes learning programming fundamentals and concepts less difficult. A significant number of people (47.6%) feel that such tools also increase their ability and skills in problem-solving for programming tasks introduced by their Computer Science program. A similar percentage of people feel that code-generation tools make learning more enjoyable.

4.1.3 Motivation, Engagement, Confidence

Regarding the increase in motivation with the use of code-generation tools, the answers are a bit ambiguous: one-third of respondents agree that the usage of code-generation tools increases motivation towards learning, another third remain neutral, and the last third either completely (9.5%) or partially disagrees (23.8%). Results from the question about the increase in student engagement were slightly different: the majority of respondents (38.1%) indicated agreement with the statement, 4.8% strongly agreed, 33.3% remained neutral, and the remaining 28.6% either completely disagreed or partially disagreed.

The survey questions about the ability to write code, in general, showed that respondents feel less confident when using code-generation tools. It is important to mention that for students who are initially confident in their programming abilities, involving code generation tools does not make a significant difference. However, for individuals who generally lack confidence in programming, it worsens when involving code generation. Thus, we observe 4.8% of people disagreeing in the first case, increasing to 19% in the second case when involving code generation. In the end, only 26.8% of respondents feel that code generation tools positively affect their confidence in programming abilities.

4.2 Interviews

Interviews were conducted with five randomly selected first-year Computer Science students as mentioned earlier. The responses revealed that all students are aware of code-generation tools and use them frequently for various learning purposes. Our exploration of their initial expectations and impressions of the use of such technologies showed that, in most cases, expectations were not high, but the results were quite impressive. Students did not anticipate code generation tools to be as intelligent and did not believe they would be as helpful.

Regarding their experience with code generation tools in the first year of the Computer Science course, all respondents stated that these tools have been excellent learning assistants and have been extensively used for coding assignments, projects, exam preparation, understanding the theoretical part of the course, and personal coding practices. Students reported that these tools are of great help when needing to create easy methods, quick functionality or logic, build components, or find minor bugs.

The students were also queried about any benefits or pitfalls related to their learning process when using code-generation tools. Among the advantages mentioned are reduced time spent on problems or bugs, as code generation tools can detect issues in the code, and improved overall learning as they effectively explain how certain things in the code work. Another positive aspect is that these tools make it easier to understand certain topics, parts of code, methods, or classes. Students mentioned that such tools provide a fresh perspective towards approaching problems. In general, students consider code-generation tools to be a great supplement to learning.

Regarding disadvantages, the main one mentioned by students is the possibility of over-reliance on the tool. Some concerns expressed by students include that it “could be bad for lazy people” or “can make people lazy.” However, all interviewees who expressed concerns acknowledged that this also depends on the individual and their intentions for using such tools.

In terms of impact on the student’s ability to develop programming proficiency, all respondents assert that code-generation tools are particularly helpful, especially for basic tasks. According to students, they increase efficiency, reduce debugging and problem-solving time, and aid in understanding concepts. Some respondents mention that while code generation tools “save” the brain from thinking, overuse can lead to a slowed development of problem-solving skills and understanding of topics.

Regarding the further integration of code generation tools into the educational realm, students’ responses suggest that such technologies should be introduced. Additionally, not only should the functionality of these tools be understood, but also how to use them most effectively should be studied. “It would not be smart to ban [code generation tools]; for many, it is a helping tool, not an answer tool,” as one respondent states. Overall, they can bring numerous advantages, including the improvement of students’ engagement in the learning process. This can occur “if all students will understand the real goal of [code generation tools] and use them in a smart way.”

5 CAPABILITIES OF AI TOOLS

This section provides a brief analysis of how ChatGPT and CoPilot perform against the coding assignments of our first and second-year courses. We’ll explore their capabilities, strengths, and limitations in handling various coding and querying tasks, ranging from basic syntax and logic to more complex algorithmic problems. This assessment aims to shed light on the practical utility of these tools in an educational setting, offering insights into their potential role in shaping the learning curve for students.

5.1 Assembly/ Low Level Coding

These were practical exercises closely related to Computer hardware. Specifically, the students worked with an Arduino and wrote assembly code for it. For certain exercises, They also were required to convert these instructions into the hexadecimal format and supply them accordingly.

Given that the problem solutions were generally small in size, ChatGPT proved to be highly useful. It competently solved the majority of the exercises presented. With knowledge of the AVR instruction set, ChatGPT was not only capable of writing the commands in assembly language but also correctly converting them to hexadecimal format, adhering to the format specified in the AVR instruction set. Additionally, it could extend specific code snippets already provided to the students by adding the requested functionality. ChatGPT could infer the appropriate registers to be used based on the sample code provided. It possessed knowledge of all essential assembly instructions required for this practical, such as ”rjmp,” ”brne,” ”nop,” ”ldi,” and so on. Throughout these exercises, ChatGPT developed code that could:

- Turn the LED ’on’ and ’off’ on the Arduino.
- Send the contents of a register to the computer via the Putty interface.
- Write a program capable of flashing a Morse code pattern for ”SOS” (this was the Weekly Assignment).

However, it has certain limitations when it comes to specific hardware components. For instance, while it can write code that utilizes the LED integrated into the Arduino board, it produces incorrect output when working with an externally connected buzzer.

5.2 Algorithms

These exercises were dedicated to exploring traditional imperative programming. Specifically, the students delved into the Python programming language and implemented widely used algorithms such as Linear Search, Bubble Sort, and Merge Sort. Throughout this process, they gained familiarity with fundamental concepts like variables, functions, and file reading, as well as slightly more advanced concepts such as recursion.

Given the simplicity of these exercises, ChatGPT and Copilot proved to be valuable resources capable of solving a majority of the exercises. The provided manual often included pseudocode to guide the students in understanding the expected algorithm structure and supplying this pseudocode to ChatGPT resulted in algorithms that adhered closely to the desired structure. While one could generate these algorithms using Copilot within their integrated development environment (IDE), the implementation may exhibit slight variations.

5.3 Databases and SQL

These exercises focused on teaching students about Database Management Systems. The students explored key theoretical concepts related to databases, including foreign keys, information querying, and creating and populating databases with data.

One type of exercise involves retrieving data from the database using simple SQL queries. For this task, ChatGPT proves to be the most helpful tool. Copilot does not perform as well since it lacks the context of the database structure, and providing it with this information is not straightforward. Before querying ChatGPT with SQL queries, it is necessary to provide the database structure. In tables where field IDs are the same across multiple tables, the ChatGPT can detect foreign key relationships directly, eliminating the need for explicit specification in such cases. ChatGPT was able to generate queries for all exercises. These exercises often involved joining multiple tables, extracting specific fields, and occasionally utilizing aggregate functions and the GROUP BY clause.

5.4 Functional Programming

The focus of these exercises was on the paradigm of functional programming, with an emphasis on working with Haskell and the GHCI compiler. The lab exercises consisted of small implementations for various exercises.

ChatGPT proved to be a useful tool for solving these problems, successfully tackling the majority of exercises with minimal issues. It possesses knowledge of advanced Haskell functions, such as higher-order functions, Guards, and recursion. Furthermore, it can utilize previously defined functions from earlier exercises, taking the context into account. ChatGPT enforces appropriate typing for each function and is familiar with QuickCheck. Some examples of exercises it was able to solve include:

- Functions to calculate the extreme values of polynomials.
- Function to determine whether a character is lowercase or uppercase.
- Defining QuickCheck properties to ensure the implemented functions work as expected.
- Function to sum the values in a list.
- Function to compute the sum of an arithmetic sequence.
- Function to count occurrences of a phrase in a sentence.
- Function to find all dividers of certain numbers and a function to check if a given number is prime.

Our results are consistent with the findings of (Geng et al., 2023), who conducted extensive research on how ChatGPT performs with exercises in the programming language OCaml. They organized various groups of functional programming exercises, each covering a different aspect of the language. For more than half of these groups, ChatGPT was able to solve 100% of the exercises, while for the remaining groups, it maintained a high accuracy rate of over 90%.

The final assignment for this topic involved an interesting and unique case study that required an understanding of additional mathematical theory that was to be implemented via Haskell functions. The necessary theory and questions were provided through an external PDF file, and students were required to fill in their answers in a '.hs' (Haskell) file. This approach effectively countered the reliance on automated tools. Solving these exercises required a deep comprehension of the theory presented in the documents, which included tables, graphs, and images. It was challenging to provide this information to the AI models, making it difficult to obtain appropriate assistance from ChatGPT. While ChatGPT could provide help with defining simpler functions, the majority of the work and the thinking process had to be done by the learner.

5.5 Advanced Imperative Programming

Advanced Imperative programming was taught to the students over the span of 10 weeks in a software development focused module. Java was utilized in this module, and students will explore various topics, including but not limited to:

- Core imperative concepts: variables, data types, arrays, methods, and exceptions.
- Object-oriented programming: classes, inheritance, and composition.
- Socket programming concepts: Server Sockets, Sockets, Readers, and Writers.
- Implementation of applications using the Model-View-Controller (MVC) pattern.
- Multithreading concepts: synchronization, race conditions, and locks.
- Software documentation using Javadoc.
- Planning and implementation of unit and integration tests.

In this traditional coding environment, the impact of Copilot becomes evident. With a codebase that was expanding every week, Copilot leverages context inference from other files in the workspace to gain a comprehensive understanding of the code written thus far, enabling it to provide appropriate suggestions. Supplying all this data to ChatGPT would be exceedingly challenging. Copilot offers several ways to assist students during these modules:

Code Suggestions. GitHub Copilot offers intelligent code suggestions based on the context of Java code. It can infer information from different classes in the workspace, providing relevant code snippets, method calls variable declarations, and more. Often, it suggests solutions before they are even considered by the coder. It encompasses knowledge of the concepts taught in this module and provides accurate and useful suggestions. As reported in (Yetistiren et al., 2022; Dakhel et al., 2022), our research also showed that using optimal method signatures and comments greatly increases the likelihood that the code generated is what the user needs.

JavaDoc Generation. Students often needed to provide detailed documentation, including JavaDoc comments, for their code. Copilot utilizes its understanding of the codebase to automatically generate reasonably accurate and meaningful JavaDoc comments. It considers method signatures and function implementations to generate relevant JavaDoc in most cases.

External Libraries: In addition to JavaDoc generation, Copilot can aid in other documentation-related tasks. It suggests relevant code examples or explanations for specific Java concepts, APIs, or design patterns as well as usage of popular external libraries.

Testing Automation. Copilot's code analysis capabilities allow it to generate integration and unit tests. It proposes test cases, test methods, and testing frameworks tailored to the Java codebase.

Error Handling and Exception Management. Copilot also helps identify potential error scenarios and suggests appropriate exception-handling mechanisms for the coder.

However, there are still potential use cases for ChatGPT. When tackling isolated problems, it can be valuable for debugging specific code sections. Additionally, when facing difficulties in approaching a problem, seeking guidance from ChatGPT for a starting point can provide initial coding assistance, with Copilot taking over to further aid the process.

Previous research by (Ouh et al., 2023) has explored the utilization of ChatGPT (without Copilot) for solving programming exercises in Java. Their experiments primarily focused on smaller, independent exercises similar to those found on platforms like LeetCode. They did not extensively cover the more complex topics included in this particular course. Nevertheless, their findings indicated that a majority of the exercises in their Java programming course could be successfully solved using ChatGPT.

5.6 Data Science with Jupyter Notebooks

As part of an AI course, students were expected to implement several data science exercises through Jupyter Notebook files. Some of the goals of these assignments include:

- Building a Neural Network
- Constructing a Decision Tree
- Building a classifier considering certain features

Many of these exercises intertwine relevant mathematical and AI theory, making it slightly cumbersome to directly convey all the theory to ChatGPT. However, throughout these practical exercises, several programming tasks still can be completed using AI tools. Some of the simple mathematical theories that require a declaration of simple functions such as Entropy, Precision, and Information Gain can be generated using ChatGPT. Additionally, Most of the tasks related to data manipulation and utilizing external libraries such as Numpy, Pandas, and SkLearn can also be automated using Copilot. Some tasks that Copilot helped with during these assignments include splitting data into training and test sets, defining a function to measure the accuracy of models, initializing a Decision Tree Classifier with appropriate parameters, and reading data from a CSV file.

6 RESULTS AND DISCUSSIONS

6.1 Viewpoint of Students

Common uses of these tools for students include assistance when stuck on a problem, verifying answers, debugging, explaining, or improving code, and explanation of fundamentals. All respondents agreed that these tools save time while programming.

Regarding disadvantages, many respondents caution against over-reliance on these tools. Moreover, they mention that generated code is often hard to understand or lacks a clear goal, making its further usage challenging. Some students also report reduced interest in becoming proficient in fundamentals. Students suggest using code generation wisely, either for time-saving on trivial tasks or for personal code and knowledge improvement.

6.2 Capabilities of AI Tools

When it comes to the capabilities of these tools. The results indicate that a majority of the exercises can be partially or completely solved with the assistance of ChatGPT and Copilot. These tools are particularly helpful for exercises related to:

- Assembly programming
- Haskell programming
- Web programming (in Java)
- SQL queries and data manipulation
- Algorithms in various languages
- Data science using Jupyter Notebooks

When the tools cannot provide solutions, they can still serve as valuable starting points, providing much of the necessary boilerplate and starting code.

Through this research, it also becomes apparent that these software tools struggle to provide solutions when they lack the appropriate context to approach a question. Adding context that is challenging to supply to these programs can reduce their effectiveness.

Some strategies to achieve this include:

- Requiring the utilization of custom files with unconventional methods as part of the exercise solutions
- Referencing specific (unique) content from lectures within the exercises
- Integrating additional theory, such as mathematics or unique case studies, into the exercises

Furthermore, making it difficult to provide context can also act as a deterrent for students using these tools. For example, in the SQL exercises that the students worked on in our tests, they did not have to explicitly specify any foreign key relations as ChatGPT could infer them automatically due to the identical field names in different tables. However, if foreign key attributes had different names, users would need to specify each foreign key relation individually, potentially discouraging tool usage, especially when there are numerous such relations involved. This is a good example of designing exercises with the use of these tools in mind.

7 LIMITATIONS AND FUTURE WORK

Due to the multifaceted nature of the learning experience, which could not be fully studied within the scope of this research, for the students' perception we had to focus on factors such as student engagement and motivation. Consequently, the work can offer relevant insights only in relation to those factors.

Additionally, the potential for bias arises as we rely on data from surveys and interviews, where participants might provide socially desirable or potentially biased answers. Moreover, the analyzed data is of limited sample size, specific time frame, and geographical focus, which may constrain the generalization of our findings. These limitations should be kept in mind when interpreting the results of this study, and future research should aim to address them.

This research project also primarily focused on smaller, bite-sized exercises when determining the capabilities of these tools. However, many programming assignments often consist of larger projects spanning the course of a few weeks. Testing these AI models on larger and more diverse projects could yield different results and provide differing valuable insights.

8 CONCLUSIONS

In conclusion, code generation technologies are developing rapidly and it is hard to ignore their prevalence among aids for learning within Computer Science programs. They have the potential to bring benefits in the educational realm if used smartly.

From our survey, we see that a majority of students already make use of these tools and have a generally positive impression of their widespread adoption. Most students feel that these tools make it easier to grasp programming fundamentals as well as improve the overall learning experience.

The analysis conducted in this report also explored how these AI tools can be utilized to assist students in solving exercises. The findings demonstrate that a significant portion of the simple exercises can be partially or completely solved with the aid of AI tools. Additionally, this paper provided insights on designing exercises that can limit the reliance on AI tools.

REFERENCES

Al Madi, N. (2022). How Readable is Model-generated Code? Examining Readability and Visual Inspection of GitHub Copilot. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, New York, NY, USA. ACM.

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., and Santos, E. A. (2022). Programming

Is Hard – Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation.

Biswas, S. (2023). Role of ChatGPT in Computer Programming. *Mesopotamian Journal of Computer Science*.

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. (2021). Evaluating Large Language Models Trained on Code.

Dakhel, A. M., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Ming, Z., and Jiang (2022). GitHub Copilot AI pair programmer: Asset or Liability?

Dwivedi, Y. K., Kshetri, N., Hughes, L., Slade, E. L., Jeyaraj, A., Kar, A. K., Baabdullah, A. M., Koohang, A., Raghavan, V., Ahuja, M., Albanna, H., Albashrawi, M. A., Al-Busaidi, A. S., Balakrishnan, J., Barlette, Y., Basu, S., Bose, I., Brooks, L., Buhalis, D., Carter, L., Chowdhury, S., Crick, T., Cunningham, S. W., Davies, G. H., Davison, R. M., Dé, R., Dennehy, D., Duan, Y., Dubey, R., Dwivedi, R., Edwards, J. S., Flavián, C., Gauld, R., Grover, V., Hu, M.-C., Janssen, M., Jones, P., Junglas, I., Khorana, S., Kraus, S., Larsen, K. R., Latreille, P., Laumer, S., Malik, F. T., Mardani, A., Mariani, M., Mithas, S., Mogaji, E., Nord, J. H., O'Connor, S., Okumus, F., Pagani, M., Pandey, N., Papagiannidis, S., Pappas, I. O., Pathak, N., Pries-Heje, J., Raman, R., Rana, N. P., Rehm, S.-V., Ribeiro-Navarrete, S., Richter, A., Rowe, F., Sarker, S., Stahl, B. C., Tiwari, M. K., van der Aalst, W., Venkatesh, V., Viglia, G., Wade, M., Walton, P., Wirtz, J., and Wright, R. (2023). Opinion Paper: "So what if ChatGPT wrote it?" Multidisciplinary perspectives on opportunities, challenges and implications of generative conversational AI for research, practice and policy. *International Journal of Information Management*, 71.

Geng, C., Zhang, Y., Pientka, B., and Si, X. (2023). Can ChatGPT Pass An Introductory Level Functional Language Programming Course?

Herrington, J. (2003). *Code Generation in Action*.

Khmelevsky, Y., Hains, G., and Li, C. (2012). Automatic code generation within student's software engineering projects. In *Proceedings of the Seventeenth Western Canadian Conference on Computing Education*, New York, NY, USA. ACM.

Li, Y., Choi, D., Chung, J., Kushman, N., Schrittwieser, J., Leblond, R., Eccles, T., Keeling, J., Gimeno, F., Dal Lago, A., Hubert, T., Choy, P., de Masson d'Autume, C., Babuschkin, I., Chen, X., Huang, P.-S., Welbl, J., Gowal, S., Cherepanov, A., Molloy, J., Mankowitz, D. J., Sutherland Robson, E., Kohli, P., de Freitas, N., Kavukcuoglu, K., and Vinyals, O. (2022). Competition-level code generation with AlphaCode. *Science*.

Marwan, S., Jay Williams, J., and Price, T. (2019). An Evaluation of the Impact of Automated Programming Hints on Performance and Learning. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*, New York, NY, USA. ACM.

Nguyen, N. and Nadi, S. (2022). An empirical evaluation of GitHub copilot's code suggestions. In *Proceedings of the 19th International Conference on Mining Software Repositories*, pages 1–5, New York, NY, USA. ACM.

Ouh, E. L., Gan, B. K. S., Jin Shim, K., and Wlodkowski, S. (2023). ChatGPT, Can You Generate Solutions for my Coding Exercises? An Evaluation on its Effectiveness in an undergraduate Java Programming Course. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, New York, NY, USA. ACM.

Ouyang, F., Zheng, L., and Jiao, P. (2022). Artificial intelligence in online higher education: A systematic review of empirical research from 2011 to 2020. *Education and Information Technologies*, 27(6):7893–7925.

Sarsa, S., Denny, P., Hellas, A., and Leinonen, J. (2022). Automatic Generation of Programming Exercises and Code Explanations using Large Language Models.

Surameery, N. M. S. and Shakor, M. Y. (2023). Use Chat GPT to Solve Programming Bugs. *International Journal of Information technology and Computer Engineering*, (31):17–22.

Wermelinger, M. (2023). Using GitHub Copilot to Solve Simple Programming Problems. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, New York, NY, USA. ACM.

Yetistiren, B., Ozsoy, I., and Tuzun, E. (2022). Assessing the quality of GitHub copilot's code generation. In *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering*, New York, NY, USA. ACM.