



# Teaching Transformed

*The apparent ability of LLMs to write functioning source code has caused celebration over the potential for massive increases in programmer productivity and consternation among teachers.*

**A**S OWNER OF GitHub and lead investor in OpenAI, the developer of the GPT-x series of large language models (LLMs), it did not take long for Microsoft to see the potential for collaboration between the two. Three years ago, GitHub partnered with OpenAI to develop Copilot as an automated assistant for programmers, quickly followed by the Copilot code-completion tool. The public release of ChatGPT by OpenAI toward the end of 2022 made the technology even more widely available to software developers and people learning to program, with other vendors joining in the effort to automate the job of writing software using LLMs.

Rapid scaling has enabled major improvements in the ability of artificial intelligence (AI) to turn natural-language requests into working code. Workplace studies have claimed LLMs boost productivity on real-world projects. In its own survey of usage by close to a million users over the year since the launch of Copilot, GitHub claimed developers accepted on average 30% of the tool's code suggestions and that usage of the suggestions increases over time as programmers become more familiar with the tool's recommendations.

For its own tests, management consultancy McKinsey recruited around 40 developers working in-house across the U.S. on two types of LLMs fine-tuned on coding problems. The experiment showed the tools could halve the time to write new code, with an average speed-up of around a third.

Though employers might welcome the productivity improvements LLMs promise, educators are concerned about how they might impact their ability to test students' understanding. Carnegie Mellon University post-doctoral fellow Jaromír Šavelka and colleagues analyzed the difference between GPT-3, which formed the ba-



sis for ChatGPT, and GPT-4, released spring 2023, on their ability to answer questions from three different Python programming courses. Earlier generations scored below 70% on even entry-level modules and would have failed the courses were they human results. GPT-4 scored 80% or higher on the three courses and would have passed.

All the instructors interviewed in a survey conducted early in 2023 by Sam Lau and Philip Guo, researchers at the University of California at San Diego, mentioned the potential for cheating as being the primary reason they would change courses followed by the arrival of tools like ChatGPT and Copilot.

Despite the advances LLMs have made, studies have shown LLMs as large as GPT-4 still have significant limitations when it comes to understanding requests and delivering suitable code. This seems likely to reduce the potential for cheating in education and for automating away jobs in the workplace. Though the McKinsey survey of its staff found improvements on simpler tasks, time savings shrank to less than 10% on tasks that developers deemed to be more difficult, or where they were trying to use a programming framework with which they were less familiar. Sometimes, tasks took junior developers up to 10% longer with the tools than without them.

An experiment by a team working at Chang'an University, China, and Griffith University, Australia, that used CodeWars katas as a suite of tests found GPT-4 could complete more than its predecessor, but failed on all the tasks in the top three levels of the eight-level contest. The models showed difficulty with optimization and simplifying algebraic equations. In one kata, the problem asked for the total area covered by a group of rectangles. GPT-4 and its predecessor could pass simple test cases, but failed to finish before the timeout deadline on the full set of tests.

Complexity is not the only barrier. Other studies have shown that LLMs can be tripped up by the order of right and wrong answers in multiple-choice questions and the way short code snippets are presented. In one example, GPT-4 failed to notice that a string-replacement function would replace all instances of the string listed in the question, but when asked about the function, it correctly explained how it works.

Students appear to learn quickly about AI's shortcomings, despite instructors' fears of them becoming overly reliant on LLMs. In a series of interviews with students, Cynthia Zastudil and colleagues at Temple University, Philadelphia, found a common complaint lay in the black-box nature of the commercial LLMs. The students often could not determine why the AI provided them with the response it did. "When using them, I don't really fully understand what they're telling me," one student said to the interviewers.

A joint study by teams from Abilene Christian University, University College Dublin, and an Auckland, NZ-based group led by Paul Denny (see the related research article in this issue, p. 56) has informed the work of a generative-AI working group<sup>a</sup> set up by the organizers of the Conference on

<sup>a</sup> <https://iticse23-generative-ai.github.io>

Innovation and Technology in Computer Science Education (ITiCSE). The experiment uncovered types of behavior that point to students developing a healthy distrust of LLMs' answers. One behavior they called "shepherding," where the users try different variations of prompts to the AI to nudge its output closer to what they want. The other, "drifting," is when students try different variants of prompts and responses before deleting the results entirely.

"Struggling students can drift when they don't have a clue what's going on. But good students can drift too. They are testing, poking, exploring," says Brett Becker, assistant professor at University College Dublin.

The immediate question for educators as these tools evolve is whether to resist AI-enabled cheating by designing tests that LLMs today find difficult to rework the syllabus to incorporate them into courses and assessments. Instructors in the surveys by Lau and Guo split almost evenly into these two groups. Some saw a need to move to face-to-face interviews to assess skills or have students perform at least some of their assignments in a classroom environment where teachers can restrict access to LLMs, though these impose a significant overhead on instructors compared to problem sheets that today are often graded automatically.

In his talk as part of the closing keynote at the 2023 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE) in July, Denny said, "Very early on, a lot of the discussion was focused on the negative aspects. We've seen headlines about terrified professors, worrying that the students are going to be cheating endlessly with these tools. Now, the narrative has changed slightly. Maybe we should try to learn how to teach more effectively with them."

A series of papers published over the past couple of years has shown LLMs can perform tasks such as identifying programming concepts in unannotated source code, and describing how they work. This kind of work would make it easier for instructors to keep up with changes in languages and applications that industry expects and reduce reliance on potentially outdated material. Becker says similar technology would help overcome a major problem

## Students are more comfortable asking an LLM for explanations than talking to a professor or teaching assistant.

for novice students: how to interpret the often-cryptic error messages provided by compilers and other software tools.

The interviews by Zastudil and colleagues indicated there is a possible mismatch between student and instructor expectations that LLMs could help fill. Though both sides saw problems with tests that appear to students as being just busywork, students felt their instructors missed a more fundamental issue the AIs could address: Traditional course materials are not always helpful for understanding underlying concepts. They find they are more comfortable asking an LLM for explanations than talking to a professor or a teaching assistant, particularly for basic concepts.


Courses that incorporate LLMs already have started, though the current offerings are experimental to differing degrees. One example at the University of California at San Diego (UCSD) started in the fall of 2023. Leo Porter, a teaching professor of computer science, is developing the course and a related book with Daniel Zingaro, associate professor in computing science at the University of Toronto.

An option some educators are pursuing is to use only a specialized LLM that is prevented from supplying compilable code as part of its answers, in the hope this will prevent students just copying and pasting the output. The UCSD course will provide access to Copilot, which runs the risk of providing material that does not directly fit the course. "We need to teach students how to determine what the code does, even if that code may be more complex than what they've been taught in class," says Porter.

One possible issue with incorporat-

ing LLMs and prompt engineering into courses is the rapid pace of development that may make some techniques quickly redundant.

"We suspect the models will improve, but the way of interacting with the models won't change dramatically. We'll hopefully see correct code generated more frequently, but we don't see the models being able to generate correct code if the person writing the prompts is ambiguous about what they want," Porter notes.

Work continues to merge LLMs with other tools to improve their ability to check the correctness of their outputs by themselves. A little more than a decade ago at the Alan Turing Centenary Conference in Manchester, U.K., Tony Hoare, emeritus professor of computer science at Wolfson College, Oxford, argued for a change to the classic Turing test. This would ask the AI not to pretend to be human, but instead to reason about its own programming. As the industry struggles to determine how well LLMs can handle the logic of programs, the need for that kind of test is becoming increasingly urgent. 

### Further Reading

Denny, P. et al.

Computing education in the era of generative AI. *Communications* 67, 2 (2024); 56–67.

Šavelka, J., Agarwal, A., Bogart, C., and Sakr, M. Large language models (GPT) struggle to answer multiple-choice questions about code. In *Proceedings of the 15th Intern. Conf. on Computer Supported Education*; DOI: 10.5220/0011996900003470

Lau, S. and Guo, P.J.

From "ban it till we understand it" to "resistance is futile": How university programming instructors plan to adapt as more students use AI code generation and explanation tools such as ChatGPT and GitHub Copilot. In *Proceedings of the 19th ACM Conf. on Intern. Computing Education Research* (2023); DOI: 10.1145/3568813.3600138

Deniz, B.K. et al.

Unleashing developer productivity with generative AI, McKinsey & Company (2023); <https://mck.co/44uig93>

Prather, J. et al.

ITiCSE working group report and student guide; <https://iticse23-generative-ai.github.io>

Chris Edwards is a Surrey, U.K.-based writer who reports on electronics, IT, and synthetic biology.

© 2024 ACM 0001-0782/24/2