Educational Opportunities and Challenges of AI Code Generation

By: Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, Eddie Antonio Santos

Presented by: Leon Lee

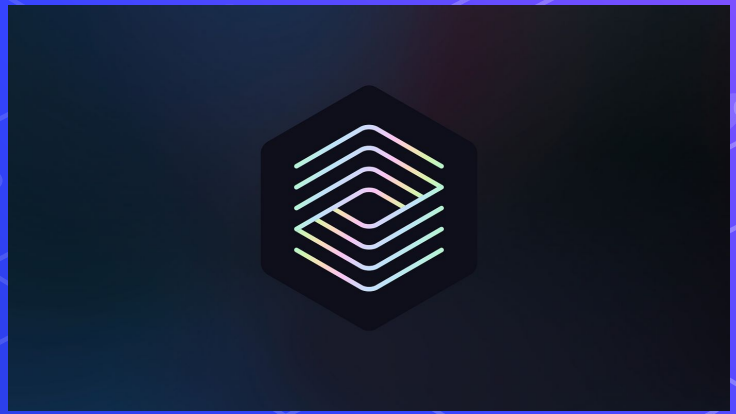
Abstract

- The recent emergence of AI-driven code generation tools presents opportunities and challenges to introductory CS courses and how they should be taught in the future
- It is imperative that the CS education community recognizes the profound impact these tools have, and act accordingly to adapt their pedagogy
- This paper serves the purpose to **spark the discussion on how AI tools will shape the future of CS pedagogy**

The Big Players: Codex, AlphaCode, CodeWhisperer

OpenAi Codex

- Trained using over 50 million Github repositories, totalling 159 GB of data
- Able to generate code in JavaScript, Go, Perl, PHP, Ruby, Swift, TypeScript, and shell, when given an English prompt
- Powers Github Copilot; a tool that provides code autocompletion within popular IDEs and it available for free to verified students and teachers
- A report conducted by Finnie-Ansley et al found that the Codex outperforms 80% of introductory CS students pn summative exam questions
- Widely available to the public and easily accessible



DeepMind AlphaCode

- Writes computer programs at a competitive level; Ranked in the top 54% in over 5000 competitive programming competitions
- Trained using over 715 GB of Github code, with generated and duplicate code filtered out from the training data
- Estimated to have perform at around the top 28% margin of users who participated in competitions in the last 6 months
- Not currently widely available

Amazon CodeWhisperer

- **Marketed as a “Machine-Learning Powered Coding Companion”**
- CodeWhisperer generates code recommendation based on a developer’s comments and prior code
- Based of off comments, it will attempt to find suitable cloud services and public libraries, and generate suggested code within the IDE
- Trained on public data
- Available to the public, but there is a waitlist, not as accessible as Codex

Opportunities with AI Code Generation

4.1: Providing Code Solutions

- AI tools excel at common algorithms and can reliably generate code such as insertion sort and tree traversals. This can **provide instructors a low cost way to produce sample solutions** and answer keys
- AI tools are able to generate a variety of solutions for the same problems or students to reference, which is important when tackling non-trivial problems
- Currently, AI tools are not concerned with the quality or style of code it generates, as it primary concern is about correctness. This may prove useful for generating discussions about quality of code when comparing different solutions

4.2: Producing Learning Resources

- The Codex AI tool has shown that is able to generate basic exercise problems for beginners, whilst offering reasonable explanation and sample solutions
- Furthermore, Codex is also capable of providing detailed step by step walkthroughs and code explanations for higher, more complex problems, most of which are sensible and correct
- Current AI models are able to illustrate fundamental computer science concepts such as common data structures, algorithms, and how to visualize a problem, which can prove to be very beneficial to students in introductory CS courses

4.3 New Pedagogical Approaches

- Instructors are able to divert time away from focusing on teaching syntax to teaching how to problem solve. AI models can be used solve low level implementation tasks, **allowing students to think of higher level algorithms and solutions**
- Clearly explaining a prompt to an AI greatly affects the success of it being able to generate a correct response. This can **encourage students to improve their ability to communicate algorithmic problems more clearly**
- Code generation and get students started on their assignments by **providing meaningful starter code and provide helpful suggestions**, which alleviates writers block of students
- Codex is very effective at explaining compilation and syntax errors, and can alleviate many menial barriers that beginner students often face

Issues with AI Code Generation

5.1 Ethical Issues

- There are **majors concerns with Academic Integrity**, primarily that students are able to outsource their assignments to AI, and it is difficult to detect cheating when this happens
- The lines between what is acceptable an acceptable practice begins to blur. It is hard to distinguish between what is generated by an IDE's support code and completely machine generated code. This also raises issues on intellectual property and licensing
- AI models take a lot of resources and computing power to train somewhere around thousands of petaflops during the pre-training process, and this raises concerns on the sustainability to develop such tools

5.2 Bias and Bad Habits

- **Questionable appropriateness for beginners**
  - AI machines are trained using public data, and it might not always be the best fit for beginners depending on specific context
- **Harmful Biases**
  - Developer found that code generation models are susceptible to harmful biases, and can generate code that reflect harmful stereotypes about gender, race, and class
  - Codex has been found generating denigratory racist comments in code comments, which is very problematic
- **Security**
  - Code generated by machine isn't guaranteed to be secure, and there is currently no way to verify that it is

5.3 Over-reliance

- Students become reliant on tools such as embed-support, and do not come up with solutions themselves
- A recent analysis of code generated by AlphaCode revealed that **11% of its Python solutions had syntax errors** and **35% of its C++ solutions did not compile**
- AI Code Generation tools are prone to making mistakes, ranging from syntax errors, incorrect/inefficient logic, and other bad coding practices
- The solutions it suggests might be appear functional, but not work as intended

Concluding Statements

Author’s thoughts and conclusion

- The impact of AI models are widespread, profound, and permanent, and it we should make quick concerted efforts to adjust the way we teach CS starting from the introductory level to accommodate these new opportunities and challenges
    - The paper suggests that CS pedagogy shifts from the focus away from code generation into code reading and evaluating
- The rise of code generation also sparks discussions and debates about the ethical implications of such tools, and what is acceptable going forward

End of presentation