The Implications of Large Language Models for CS Teachers and Students

Stephen MacNeil
Temple University
Philadelphia, PA, USA
stephen.macneil@temple.edu

Joanne Kim
Temple University
Philadelphia, PA, USA
joanne.kim@temple.edu

Juho Leinonen
Aalto University
Espoo, Finland
juho.2.leinonen@aalto.fi

Paul Denny
The University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Seth Bernstein
Temple University
Philadelphia, PA, USA
seth.bernstein@temple.edu

Brett A. Becker
University College Dublin
Dublin, Ireland
brett.becker@ucd.ie

Michel Wermelinger
The Open University
Milton Keynes, UK
michel.wermelinger@open.ac.uk

Arto Hellas
Aalto University
Espoo, Finland
arto.hellas@aalto.fi

Andrew Tran
Temple University
Philadelphia, PA, USA
andrew.tran10@temple.edu

Sami Sarsa
Aalto University
Espoo, Finland
sami.sarsa@aalto.fi

James Prather
Abilene Christian University
Abilene, TX, USA
james.prather@acu.edu

Viraj Kumar
Indian Institute of Science
Bengaluru, India
viraj@iisc.ac.in

ABSTRACT

The introduction of Large Language Models (LLMs) has generated a significant amount of excitement both in industry and among researchers. Recently, tools that leverage LLMs have made their way into the classroom where they help students generate code and help instructors generate learning materials. There are likely many more uses of these tools – both beneficial to learning and possibly detrimental to learning. To help ensure that these tools are used to enhance learning, educators need to not only be familiar with these tools, but with their use and potential misuse. The goal of this BoF is to raise awareness about LLMs and to build a learning community around their use in computing education. Aligned with this goal of building an inclusive learning community, our BoF is led by globally distributed discussion leaders, including undergraduate researchers, to facilitate multiple coordinated discussions that can lead to a broader conversation about the role of LLMs in CS education.

CCS CONCEPTS

• **Social and professional topics → Computing education**.

KEYWORDS

large language models, GPT-3, code explanations, code generation, copilot, artificial intelligence, computer science education

ACM Reference Format:
Stephen MacNeil, Joanne Kim, Juho Leinonen, Paul Denny, Seth Bernstein, Brett A. Becker, Michel Wermelinger, Arto Hellas, Andrew Tran, Sami Sarsa, James Prather, and Viraj Kumar. 2023. The Implications of Large Language Models for CS Teachers and Students. In *Proceedings of the 54th ACM Technical Symposium on Computing Science Education V. 2 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3545947.3573358

1 INTRODUCTION

The recent introduction of Large Language Models (LLMs) is beginning to provide exciting new opportunities for professional computer scientists and for computing educators. For example, LLMs power tools like GitHub Copilot, Amazon CodeWhisperer, and tabnine which can automatically generate code based on natural language specifications provided by a user [1, 3, 4, 11]. LLMs are also capable of generating high-quality explanations of code in real-time to facilitate learning [7, 9, 10]. Recent work has even demonstrated their potential for generating assignments for instructors [10] and enhancing error messages [6]. However, the introduction of LLMs presents both opportunities and challenges [2]. These generative models are very new, however, and it is not yet clear to what extent they can facilitate learning in practice.

In this Birds of a Feather session, we plan to engage the SIGCSE community in fruitful discussions about whether and how these new LLMs might transform our research and practice. With the growing interest in LLMs in CS education over the past six months [2, 5–10, 12, 13], we expect that this is a critical moment for our community to engage with these new opportunities while critically reflecting on the challenges of the present moment. Our goal is to build a learning community of computing educators that can sustain these conversations

1.1 A changing workplace

Developer tools are constantly improving and enabling developers to work at increasingly high levels of abstraction. LLMs that power IDE plugins such as GitHub Copilot[1], and services such as Amazon CodeWhisperer[2] are poised to have an even greater impact based on

[1]github.com/features/copilot
[2]aws.amazon.com/codewhisperer/

their ability to not only facilitate developers' work but to actually do work on their behalf. These new capabilities prompt CS education researchers to reflect critically on the skills that students should be learning to work effectively in these environments. Echoing calls for computing skills beyond simply coding, such as computational thinking and computing ethics, CS education research may need to re-evaluate our curriculum and intended learning outcomes.

1.2 A changing classroom

Large Language Models are currently poised to transform computer science classrooms in a variety of ways. Recent papers have started to illustrate ways that LLMs might impact students' experiences [5], teachers' experiences [10], and provide new learning opportunities [7, 9]. While these changes ideally benefit students and teachers, there have also been concerns raised about the effects LLMs might have on academic integrity [5]. This moment is vital for computing education researchers to speculate, discuss, and define the ways that LLMs should be introduced to students. With care, decisions made now can ensure that LLMs are used in ways that enhance teaching and learning rather than introduce challenges.

2 ORGANIZERS

Collectively, our team has presented six papers, posters, and workshops related to the use of LLMs in CS education [5, 7–10, 13], including classroom deployments [7]. As organizers, we each bring additional perspectives coming from seven universities (including distance education) across three different continents. Most importantly, our team includes three undergraduate researchers and one Ph.D. researcher. We strongly believe that this student perspective is essential for research that involves students engaging with LLMs.

2.1 As instructors, we...

As instructors, we care about creating high-quality learning environments for our students. By using LLMs to generate real-time explanations [7, 9, 10] and programming assignments [10], instructors can quickly and cheaply generate learning materials to supplement existing teaching practice, freeing instructors to spend time with students who require more help or refining their existing materials. This ability to generate on-demand learning resources also provides opportunities to personalize content for each individual student. However, we are also concerned that LLMs have the potential to do students' work for them. Students may be tempted to use tools like GitHub Copilot to complete assignments early in their program, but may not be able to rely on them as the assignments become more challenging. Our team consists of seven educators with a wide range of experience levels which inspired the following topics:

- Students may eventually use LLMs to generate code at work, what does this mean for curriculum design?
- LLMs can generate code comparable to code written by students [10], what does this mean for academic integrity?
- LLMs can generate learning materials such as assignments [10] and explanations [7, 9], in what other ways can LLMs support pedagogy and teaching?

2.2 As students, we...

As students, the challenges and opportunities we face are primarily around learning and getting full-time positions. Some of us have taken high school courses and are sufficiently prepared, whereas some of us might need more help. LLMs offer personalized content which can be accessed on demand at any time. This could potentially reduce our hesitation to reach out for help and also help increase students' confidence. In addition, when transitioning into the professional environment, LLMs may be tools that we rely on to do our work. If GitHub Copilot can reduce our professional workload, why wouldn't we use it? Our discussions have inspired the following topics:

- Explanations are sometimes inaccurate or incomplete. What effect will wrong explanations have on student learning?
- Crafting prompts for LLMs can be challenging, what courses would be available to teach students to use LLMs?
- Generated code may be incorrect. What effect does this have on learning?

3 MULTIPLE BREAKOUT ROOM DISCUSSIONS

This BoF will be an online event, so that it reaches more attendees, given the expected interest in this topic. We will facilitate discussions in up to six breakout rooms, with one discussion topic per room. There will be two discussion rounds so that attendees can switch rooms and discuss two topics. Rooms that contain student researchers will focus on topics where student perspectives are most crucial such as designing new courses or learning activities.

- 5 Minutes - Breakout room topics are introduced and attendees asynchronously introduce themselves in the chat.
- 20 Minutes - First round of guided discussions
- 20 Minutes - Second round of guided discussions
- 15 Minutes - Share back, summarize insights, and brainstorm topics for future discussion.

If accepted, our team will create a website to advertise for the BoF, solicit additional topics of interest, and upload a post-BoF report based on the notes taken by breakout room discussion leaders. In this way, the BoF will serve as the starting point for broader conversations within our community about LLMs. We hope that attendees will return to this website over time as we add additional events and best practices as they relate to LLMs.

REFERENCES

[1] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2022. Grounded Copilot: How Programmers Interact with Code-Generating Models. https://arxiv.org/abs/2206.15000
[2] Brett Becker, James Prather, Paul Denny, Andrew Luxton-Reilly, James Finnie-Ansley, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities And Challenges of AI Code Generation. In *Proc. SIGCSE '23* (Toronto, Canada). ACM.
[3] Mark Chen, Jerry Tworek, Heewoo Jun, et al. 2021. Evaluating Large Language Models Trained on Code. https://doi.org/10.48550/ARXIV.2107.03374
[4] Paul Denny, Viraj Kumar, and Nasser Giacaman. 2023. Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems using Natural Language. In *Proc. SIGCSE '23* (Toronto, Canada). ACM.
[5] James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Proc. Australasian Computing Education Conf.* ACM, 10–19. https://doi.org/10.1145/3511861.3511863
[6] Juho Leinonen, Brett A. Becker, Paul Denny, Arto Hellas, James Prather, Brent Reeves, and Sarsa Sami. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proc. SIGCSE '23* (Toronto, Canada). ACM.

[7] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proc. SIGCSE'23*. ACM, 6 pages.

[8] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Generating CS Learning Materials with Large Language Models. In *Proc. SIGCSE'23*. ACM, 3 pages.

[9] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating Diverse Code Explanations Using the GPT-3 Large Language Model. In *Proc. ICER'22 - Vol. 2*. ACM, 37–39.

[10] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proc. ICER'22 - Vol. 1*. ACM, 27–43.

[11] Dominik Sobania, Martin Briesch, and Franz Rothlauf. 2021. Choose Your Programming Copilot: A Comparison of the Program Synthesis Performance of GitHub Copilot and Genetic Programming. https://arxiv.org/abs/2111.07875

[12] Justin D Weisz, Michael Muller, Steven I Ross, Fernando Martinez, Stephanie Houde, Mayank Agarwal, Kartik Talamadupula, and John T Richards. 2022. Better together? An Evaluation of AI-supported Code Translation. In *Proc. 27th Int'l Conf. on Intelligent User Interfaces*. ACM, 369–391.

[13] Michel Wermelinger. 2023. Using GitHub Copilot to Solve Simple Programming Problems. In *Proc. SIGCSE'23*. ACM, 6 pages.