ABSTRACT
The recent advent of highly accurate and scalable large language models (LLMs) has taken the world by storm. From art to essays to computer code, LLMs are producing novel content that until recently was thought only humans could produce. Recent work in computing education has sought to understand the capabilities of LLMs for solving tasks such as writing code, explaining code, creating novel coding assignments, interpreting programming error messages, and more. However, these technologies continue to evolve at an astonishing rate leaving educators little time to adapt. This working group seeks to document the state-of-the-art for code generation LLMs, detail current opportunities and challenges related to their use, and present actionable approaches to integrating them into computing curricula.

CCS CONCEPTS
• Social and professional topics → Computing education; • Computing methodologies → Artificial intelligence.

KEYWORDS
AI; artificial intelligence; code generation; Codex; computer programming; Copilot; CS1; GitHub; GPT; large language models; LLM; novice programming; OpenAI; pedagogical practices
1 INTRODUCTION

Recent advancements in artificial intelligence (AI) have ushered in a new era of computing. One particular class of AI models, known as large language models (LLMs), has shown remarkable capabilities in the generation and interpretation of natural language data and source code interpretation and generation. Many timely and important questions remain unanswered about how we will adapt to the challenges and opportunities this presents. If students are able to generate solutions to all of their programming coursework, how does this impact what is taught, how it is taught, and how students will remain motivated to learn?

For instance, many introductory programming courses use a popular evidence-based approach involving students writing dozens of small programming exercises checked by automated assessment tools. These problems can now be solved quite easily by new tools that provide students access to powerful LLMs [3]. The most recent models can solve even more complex data structures and algorithms-level assignments [4]. This casts doubt about the efficacy and longevity of current pedagogical practices and raises concerns about student learning, plagiarism, and over-reliance [1]. Current tools, such as GitHub Copilot, can provide code solutions in a student’s IDE and are free to use. However, for every right answer these tools provide, they also can provide wrong or ambiguous code and include unnecessary elements [11]. Students using these tools can also quickly become lost while reading code they didn’t write [5] or lazily drift from code suggestion to code suggestion without understanding what they are doing [9].

The emergence of large language models could also bring benefits, however. These models can be used to scaffold instructors in creating educational resources such as novel, personalized programming exercises [2,10], code explanations that could help support students when they are working on exercises [6,8], and enhanced programming error messages that might be easier to understand for novice programmers [7].

We believe that large language models will have profound impacts on computing education. This working group will work towards understanding how we can make those impacts as positive as possible.

2 GOALS

This working group is motivated by the following goals:

(1) Identify areas/aspects of computing education where LLMs could be used from both the student and teacher perspective. We plan to collect data from a wide range of computing educators to identify and expand upon the most influential candidate areas.

(2) Present a guide to the opportunities and challenges of LLMs in computing education as well as likely future opportunities and challenges, given that these models seem likely to improve rapidly.

(3) Replicate prior work on the performance of current LLM models on programming problems, exam questions, and other curricula. From this work, choose appropriate benchmarks by which future work can determine the efficacy of these models and provide a standard for replication.

(4) Create an evidence-based resource of pedagogical approaches for which LLMs can be utilised so that programming educators can utilise these new tools effectively.

(5) Cast a bold yet practical vision for the future of programming education in this new era.

Although we plan to include an extensive review of the literature, we recognize that any such attempt in this nascent and rapidly expanding area of research will quickly become out of date. We will therefore focus on the present status questionis and provide recommendations based on that.

REFERENCES


