Beyond the Textbook: Rethinking Students‘ Competencies in the LLM Era

Dr. Natalie Kiesler

Senior Researcher

Educational Computer Science Group

DIPF | Leibniz Institute for Research and Information in Education

November 13, 2023 | Uppsala University, Sweden

Event: Generative AI: Implications for Teaching and Learning

Agenda

1.) Introduction

2.) Related Work

3.) Performance of Large Language Models in Introductory Programming Tasks

4.) Feedback Provided by Large Language Models

5.) Implications on Students‘ (and Educators‘) Competencies

6.) Discussion and Conclusions

7.) Future Work

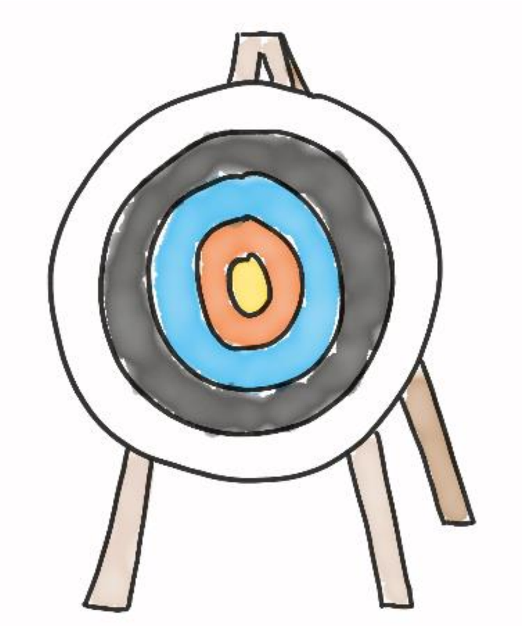
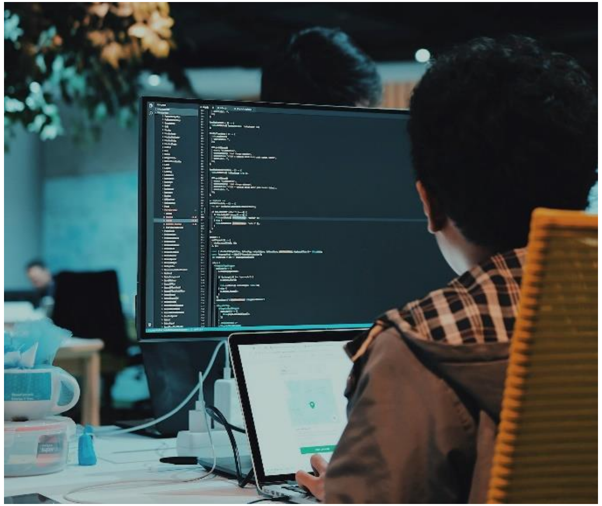


Illustration: Alexander Sperl

Introduction

- Challenges in Introductory Programming Education are manifold
  - Increasing student numbers
  - Limited staff resources
  - Heterogeneous learner groups
  - Need for formative feedback
- Growing demand for automated teaching support, while current learning environments are limited (Jeuring et al. 2022)
- Introduction of ChatGPT in late November 2022 as easily accessible, powerful tool
- Promising performance, code explanation, and error analysis in introductory programming tasks (Finne-Ansley et al. 2022, 2023)

➔ Need to explore the potential of LLMs in Computing Education



Related Work

LLMs have become subject to research, e.g.,

- Performance of OpenAI’s Codex in CS 1 and CS 2 tasks (Finnie-Ansley et al. 2022, 2023)
- Educational opportunities and challenges of LLMs (Becker et al. 2023)
- Investigating student learning with and without LLMs (Kazemitabaar et al. 2023)
- Application of LLMs to generate exercises (Sarsa et al. 2022) and code explanations (MacNeil et al. 2022)
- Exploration of prompt engineering, how to phrase questions (Denny et al. 2023)
- ITiCSE working group on curiccula, learning objectives, pedagogy, ethics, educators‘ and students‘ perceptions, and many other aspects (Prather et al. 2023)

➔ There are still many open questions, and few classroom experiences!

Performance of LLMs in Introductory Programming Tasks

DIPF

Goals and Research Questions:

- Exploring the performance of ChatGPT-3.5 (May 24) and GPT-4 when solving introductory programming tasks; and discuss implications for higher education and assessments
- RQ: How does an LLM like ChatGPT solve introductory programming tasks?

Method:

- Selection of all 72 CodingBat Python tasks, Requesting ChatGPT (3.5 and GPT-4) to generate a solution (by using the task and input)
- Evaluating the outputs‘ quality via CodingBat (i.e., unit tests)

Results:

- Summary of ChatGPT-3.5‘s and GPT-4‘s performance in 72 introductory programming tasks (from CodingBat)

Performance of LLMs in Introductory Programming Tasks

python Copy code

```python
def alarm_clock(day, vacation):
    # Check if on vacation
    if vacation:
        # If on vacation, check if weekend
        if day == 0 or day == 6:
            return 'off'  # 'off' on weekends during vacation
        else:
            return '10:00'  # '10:00' on weekdays during vacation
    else:
        # If not on vacation, check if weekend
        if day == 0 or day == 6:
            return '10:00'  # '10:00' on weekends
        else:
            return '7:00'  # '7:00' on weekdays
```

Performance of LLMs in Introductory Programming Tasks

DIPF

Given a day of the week encoded as 0=Sun, 1=Mon, 2=Tue, ...6=Sat, and a boolean indicating if we are on vacation, return a string of the form "7:00" indicating when the alarm clock should ring. Weekdays, the alarm should be "7:00" and on the weekend it should be "10:00". Unless we are on vacation -- then on weekdays it should be "10:00" and weekends it should be "off".

alarm_clock(1, False) → '7:00'
alarm_clock(5, False) → '7:00'
alarm_clock(0, False) → '10:00'

Go ...Save, Compile, Run (ctrl-enter)

```
def alarm_clock(day, vacation):
    if vacation:
        if day >= 1 and day <= 5:  # Weekdays
            return '10:00'
        else:  # Weekends
            return 'off'
    else:  # Not on vacation
        if day >= 1 and day <= 5:  # Weekdays
            return '7:00'
        else:  # Weekends
            return '10:00'
```

Expected	Run		
alarm_clock(1, False) → '7:00'	'7:00'	OK	
alarm_clock(5, False) → '7:00'	'7:00'	OK	
alarm_clock(0, False) → '10:00'	'10:00'	OK	
alarm_clock(6, False) → '10:00'	'10:00'	OK	
alarm_clock(0, True) → 'off'	'off'	OK	
alarm_clock(6, True) → 'off'	'off'	OK	
alarm_clock(1, True) → '10:00'	'10:00'	OK	
alarm_clock(3, True) → '10:00'	'10:00'	OK	
alarm_clock(5, True) → '10:00'	'10:00'	OK	
other tests		OK	

All Correct

next | chance

Python > Logic-1

done page

Code is saved so long as this session is active. Create an account above to save code past this session.

Performance of LLMs in Introductory Programming Tasks

CodingBat task area	GPT-3.5 textual explanation	GPT-3.5 program code	GPT-3.5 correct unit test results	GPT-4 textual explanation	GPT-4 program code	GPT-4 correct unit test results
Warmup-1	11/12	12/12	12/12	12/12	12/12	12/12
Warmup-2	9/9	9/9	9/9	9/9	9/9	9/9
String-1	11/11	11/11	10/11	11/11	11/11	11/11
List-1	12/12	12/12	12/12	11/12	12/12	12/12
Logic-1	8/9	9/9	8/9	9/9	9/9	9/9
Logic-2	7/7	7/7	6/7	6/7	7/7	6/7
String-2	6/6	6/6	6/6	6/6	6/6	4/6
List-2	6/6	6/6	6/6	6/6	6/6	5/6

Results (excerpt):

- GPT-3.5 solves 69 of 72 tasks on the first attempt (95,5%), GPT-4 68 of 72 tasks (94,4%) (Kiesler and Schiffner 2023)
- Problems where the LLM did not succeed on first attempt contained, e.g., syntactic ambiguity
- Disclaimer: LLMs may have been trained on the model solutions, and tasks were well described.

Feedback Provided by Large Language Models

DIPF

Goals and Research Questions:

- Exploration of generative AI (ChatGPT as LLM) to generate formative feedback in response to students' solutions to introductory programming tasks.
- RQs: What output is generated by an LLM like ChatGPT in response to a beginner student help request? How can we characterize the output in terms of feedback?

Method:

- Selection of four tasks and submissions to weekly exercises (from week 1-4 of a CS1 class, 300 students)
- Design-based Research, iterative approach, exploration of prompts („What's wrong with my code?"), 3 regenerated answers (March 23)
- Development of inductive categories to characterize the output of the feedback

Results (excerpt):

- Overview of criteria to characterize the output generated by ChatGPT (e.g., Content, Quality, Other)

Feedback Provided by Large Language Models

DIPF

Results (excerpt):

- Table with characterization of ChatGPT's responses to students' help request with students' solution (Kiesler, Lohr, Keuning 2023)

Symbol	Meaning
INFO	Requesting more information
STYLE	Stylistic suggestion
CAUSE	Textual explanation cause of error
FIX	Textual explanation fix of error
CODE	Code provided
EXA	Illustrating examples
COMP	Code, if provided, compiles
MIS	Misleading information
UNC	Uncertainty
META	Meta-cognitive elements
MOT	Motivational elements

	CONTENT																		QUALITY									OTHER					
	INFO			STYLE			CAUSE			FIX			CODE			EXA			COMP			MIS			UNC			META			MOT		
Stud_task	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3	R1	R2	R3
01_TEOS	○	○	○	○	○	○	○	●	●	○	○	○	●	◐	●	○	○	○	●	–	●	○	●	●	○	○	○	○	○	○	○	○	○
02_TEOS	○	○	○	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	○	○	○	○	○	○	○	○	○	○
03_TEOS	○	●	○	○	○	○	●	●	●	●	●	●	◐	○	○	○	○	○	–	–	–	●	○	○	○	○	○	○	○	○	○	○	○
04_TEOS	○	○	○	○	●	○	○	○	○	○	○	○	○	○	○	○	○	○	–	–	–	○	○	○	●	●	●	●	●	●	○	○	○
05_TEOS	○	○	○	●	●	●	●	●	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○	○
06_TEOS	○	○	○	●	○	●	●	●	●	●	●	●	◐	●	◐	●	●	●	–	●	–	○	○	○	○	○	○	○	○	○	○	○	○
07_TEOS	○	○	○	○	○	○	●	●	●	●	●	●	●	●	○	○	○	○	●	●	–	○	●	○	○	○	○	○	○	○	○	○	○
08_TEOS	○	○	○	○	●	○	●	○	●	●	●	●	◐	○	○	○	○	○	–	–	–	○	○	○	○	○	○	○	○	○	●	○	○
09_TEOS	○	○	○	○	●	○	●	●	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○
10_TEOS	○	○	○	○	●	○	●	●	●	●	●	●	●	●	●	○	○	○	○	●	●	●	○	○	○	○	○	○	○	○	○	○	○
01_TTBS	○	○	○	○	○	○	●	●	●	○	●	●	○	●	●	○	●	○	–	●	●	●	○	○	●	○	○	○	○	○	○	○	○
02_TTBS	○	○	○	○	○	○	●	●	●	●	●	●	●	●	●	○	○	●	–	●	●	○	○	○	○	○	○	○	○	○	○	○	○
03_TTBS	○	○	●	○	●	●	●	○	●	●	○	○	●	○	○	○	○	○	●	–	–	●	○	●	○	○	●	○	○	○	○	○	○
04_TTBS	○	○	○	○	○	○	●	●	●	●	●	●	●	◐	●	○	○	○	●	●	–	○	○	●	○	○	○	○	○	○	○	○	○
05_TTBA	○	○	○	○	○	●	●	●	●	●	●	●	◐	●	●	○	●	○	–	●	●	○	●	●	○	○	○	○	○	○	○	○	○
06_TTBA	○	○	○	●	○	○	●	●	●	●	●	●	○	●	◐	○	○	○	–	●	–	●	●	●	●	○	○	○	○	○	○	○	○
07_TTBA	○	○	○	●	●	○	●	●	●	●	●	●	○	◐	◐	○	○	●	–	–	–	●	●	●	○	●	○	○	○	○	○	○	○
08_TTBA	○	○	○	○	○	○	●	●	●	●	●	●	●	●	●	●	○	●	●	○	●	●	●	●	○	○	○	○	○	○	○	○	○
09_TTBA	○	○	○	○	○	○	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	○	●	○	○	○	○	○	○	○	○	○
10_TTBA	○	○	○	○	○	○	●	●	●	●	●	●	●	●	●	●	○	○	●	●	●	●	●	●	○	○	○	○	○	○	○	○	○
01_NEGF	○	○	○	○	●	●	●	○	●	●	○	●	●	○	○	●	○	○	●	–	–	●	○	●	○	●	●	○	●	○	○	○	○
02_NEGF	○	○	○	○	○	●	●	●	●	●	●	●	●	●	○	○	○	○	●	●	–	●	●	●	○	○	●	○	○	○	○	○	●
03_NEGF	○	○	○	○	○	●	●	●	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●	○	○	○	○	○	○	●	○	○

Implications on Students‘ and Educator‘s Competencies

Students‘ Competencies (which we implicitly expect):

- Students need to understand the task description.
- Students need to be able to express implicit side conditions/restrictions by the educator in their own words.
- Students need to analyze and evaluate the output generated by the LLM (more than understanding!):
  - Students have to trace code.
  - Students have to predict the outcome of the generated code.
  - Students have to test the generated code.
  - Students have to evaluate the adequacy of the output w.r.t. correctness (e.g., if it compiles) and style
  - Students have to evaluate the adequacy of the output w.r.t. the task and other limitations.
- Students may have to integrate the generated solution (e.g., code snippet) into their own code.
- Students need to develop adequate (follow-up) prompts.

Implications on Students‘ and Educator‘s Competencies

DIPF

Implications on Educators‘ Competencies:

- Educators need to formulate tasks adequate for novices (without ambiguity), and transparently communicate their expectations.
- Educators need to know the basic principles of how LLMs generate textual output including code.
- Educators need to know the limitations for LLMs like ChatGPT and their (rapidly increasing) potential to solve programming tasks.

Other aspects:

- Educators need to be open to change, i.e. adapt to this new, ubiquitous tool.
- Educators need to acknowledge the existence of this tools in their classroom.
- Educators need experience in using LLMs themselves.

Discussion and Conclusions

- ChatGPT can solve well-known introductory programming tasks.
- ChatGPT has the potential to address learners' informational needs.
- ChatGPT offers more types of feedback than other learning environments (Jeuring et al. 2022).

- **BUT:**
  - The output greatly varies depending on prompts, and misleading information for novices is contained.
  - Educators need to support students using these tools, and should not ignore them.
  - Will we see a shift towards student-centered teaching, learning, and assessment?
  - Will we put more emphasis on understanding (i.e., "reading" or "talking about") code instead of checking the produced code?
  - How can we prepare students ideally to use LLMs?

Future Work

- Open research questions to address:
  - How can we reduce the misleading information provided by ChatGPT and LLMs?
  - How can we manipulate prompts to receive a certain type of feedback? (work-in-progress)
  - How can we guide students and train educators to safely use LLMs?
  - To what extent can LLMs help broaden participation in CS or programming?
  - How accessible is this technology?

Tack för din uppmärksamhet!

Literature

- **Becker et al. 2023:** Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation. In Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023). ACM, New York, USA, 500–506. https://doi.org/10.1145/3545945.3569759
- **Finnie-Ansley et al. 2022**: 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 Proceedings of the 24th Australasian Computing Education Conference (ACE '22). ACM, New York, USA, 10–19. https://doi.org/10.1145/3511861.3511863
- **Finnie-Ansley et al. 2023:** James Finnie-Ansley, Paul Denny, Andrew Luxton-Reilly, Eddie Antonio Santos, James Prather, and Brett A. Becker. 2023. My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises. In Proceedings of the 25th Australasian Computing Education Conference (ACE '23). ACM, New York, USA, 97–104. https://doi.org/10.1145/3576123.3576134
- **Denny et al. 2024:** Paul Denny, Juho Leinonen, James Prather, Andrew Luxton-Reilly, Thezyrie Amarouche, Brett A. Becker, and Brent N. Reeves. 2023. Promptly: Using Prompt Problems to Teach Learners How to Effectively Utilize AI Code Generators. https://arxiv.org/abs/2307.16364. *To appear in Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024). ACM, New York, USA.*
- **Jeuring et al. 2024:** Johan Jeuring, Hieke Keuning, Samiha Marwan, Dennis Bouvier, Cruz Izu, Natalie Kiesler, Teemu Lehtinen, Dominic Lohr, Andrew Peterson, and Sami Sarsa. 2022. Towards Giving Timely Formative Feedback and Hints to Novice Programmers. In Proceedings of the 2022 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '22). ACM, New York, USA, 95–115. https://doi.org/10.1145/3571785.3574124

Literature

- **Kazemitabaar et al. 2023**: Majeed Kazemitabaar, Xinying Hou, Austin Henley, Barbara Ericson, David Weintrop and Tovi Grossman. How Novices Use LLM-Based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment. ArXiv abs/2309.14049. https://doi.org/10.48550/arXiv.2309.14049
- **Kiesler, Lohr & Keuning, 2023:** Natalie Kiesler, Dominic Lohr, and Hieke Keuning. 2023. Exploring the Potential of Large Language Models to Generate Formative Programming Feedback. https://arxiv.org/abs/2309.00029
- **Kiesler & Schiffner, 2023:** Natalie Kiesler and Daniel Schiffner. 2023. Large Language Models in Introductory Programming Education: ChatGPT's Performance and Implications for Assessments. *arXiv preprint arXiv:2308.08572*. https://doi.org/10.48550/arXiv.2308.08572
- **MacNeil et al. 2023**: 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 (SIGCSE 2023). ACM, New York, USA, 931–937. https://doi.org/10.1145/3545945.3569785
- **Prather et al. 2023:** James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, Stephen MacNeil, Andrew Peterson, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots are Here: Navigating the Generative AI Revolution in Computing Education. https://arxiv.org/abs/2310.00658. *To appear in* Proceedings Companion of the 28th Annual ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE 2023 Companion).
- **Sarsa et al. 2022:** Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1 (Lugano and Virtual Event, Switzerland) (ICER ’22). ACM, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957