Hellas, Arto; Leinonen, Juho; Leppänen, Leo

Experiences from Integrating Large Language Model Chatbots into the Classroom

Published in:
SIGCSE Virtual 2024: Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1

DOI:
10.1145/3649165.3690101

Published: 05/12/2024

Document Version
Publisher's PDF, also known as Version of record

Experiences from Integrating Large Language Model Chatbots into the Classroom

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

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

Leo Leppänen
University of Helsinki
Helsinki, Finland
leo.leppanen@helsinki.fi

ABSTRACT

We provided students access to a state-of-the-art large language model (LLM) chatbot through the online materials of three university-level courses. One of the courses focused on software engineering with LLMs, while the two other courses were not directly related to LLMs. The chatbot used OpenAI GPT-4 without additional filters or system prompts.

Our results suggest that only a minority of students engage with the chatbot in the courses that do not relate to LLMs. At the same time, unsurprisingly, nearly all students in the LLM-focused course leveraged the chatbot. In all courses, the majority of the chatbot usage came from a few superusers, whereas the majority of the students did not heavily use the chatbot even though it effectively provided free access to OpenAI's GPT-4 model (which would have otherwise required a paid subscription at the time of the study). We observe that in addition to students using the chatbot for course-specific purposes, many use the chatbot for their own purposes.

Overall, our results suggest that the worst fears of educators – all students overrelying on chatbots – did not materialize. Finally, we discuss potential reasons for low usage, including the need for more tailored and scaffolded chatbot experiences targeted for specific types of use cases.

CCS CONCEPTS

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

KEYWORDS

large language models, generative AI, chatbots, classroom experiences, experience report, usage analysis

ACM Reference Format:
Arto Hellas, Juho Leinonen, and Leo Leppänen. 2024. Experiences from Integrating Large Language Model Chatbots into the Classroom. In *Proceedings of the 2024 ACM Virtual Global Computing Education Conference V. 1 (SIGCSE Virtual 2024), December 5–8, 2024, Virtual Event, NC, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3649165.3690101

SIGCSE Virtual 2024, December 5–8, 2024, Virtual Event, NC, USA

ACM ISBN 979-8-4007-0598-4/24/12.
https://doi.org/10.1145/3649165.3690101

1 INTRODUCTION

Large Language Models (LLMs) such as ChatGPT have captured the attention of both academia and the general public. Initial hype—especially outside of academic works—has framed LLMs as replacements for many creative and knowledge workers. Reactions to the use of LLM-based generative systems in academic settings have been mixed, ranging from calls for—and realized—bans [5] to claims that they are the new normal and teaching should be reorganized around them [6, 25].

The avalanche of LLMs is visible also in computing education and computing education research, where researchers have highlighted a variety of tasks that LLMs can do [7, 23, 33]. As students begin to use these tools, several of the threats identified by researchers have come into clearer focus. For instance, students often do not understand the code automatically generated by LLMs because they did not write it [34]. Even so, students may quickly accept incorrect code suggestions and tinker with the code before discovering they do not need it, only to start over again [14, 34, 39]. More generally, there's evidence that the use of LLM assistants may, for example, lead to programmers writing less secure code [31].

In this article, we outline our experiences from integrating a state-of-the-art LLM-powered[1] chatbot into three university-level CS courses. The integration with the LLM was unfiltered, meaning that students could also discuss contents unrelated to the courses. The closest match to our study was recently conducted by Prasad et al. [32], who provided unrestricted access to an LLM through a programming environment plugin, and explored how students used the LLM. While a lot of concerns about potential student over-reliance have been raised in previous work [33], combined with the prior study by Prasad et al. [32], our work provides further information on how students use an unrestrained LLM chatbot that is purposefully similar to commercially available LLM-based chatbots such as ChatGPT.

Our research questions for the present study are as follows: (**RQ1**) How does the use of the LLM-based course assistant relate to the course?; (**RQ2**) How does the perceived usefulness of the LLM-based course assistant relate to the course?; and (**RQ3**) How does the use of the LLM-based course assistant relate to student background, and prior experience with LLMs?

2 BACKGROUND

Computing education as a field has been continuously evolving, influenced by increases in processing power, availability of personal computers, access to the internet, online learning management systems, open online courses, and most recently large language

[1]We used GPT-4 which was the best model available at the time of the study in 2023.

models. The number of learners is huge, bolstered by initiatives such as Computer Science for All [12] and Hour of Code [11], where the latter has reported over a hundred million students.

Programming education—a part of computing education—and the ways in which programming education could be improved is a significant research topic in computing education research [22]. When learning to program, students learn to understand both syntax and semantics of the programming language, slowly acquire plans that are used to reach reoccurring goals, and learn pragmatic aspects such as working with the available tools [8]. This takes plenty of effort and can be challenging; nearly one-third of introductory programming students in higher education fail the introductory programming course [42]. Improving pedagogy, classroom design, and instruction can help improve retention, although even after improvements, there still exists a considerable body of students who fail [41].

Courses employ teaching assistants in a variety of tasks including assisting students in programming labs, grading assignments, and giving office hours [26]. With infinite qualified teaching assistants, a classroom could in principle employ one-to-one mastery learning, which has been shown to lead to two standard deviation improvement in learning outcomes when compared to students in traditional classroom [2]. However, the increasing enrollments in programming classrooms and the associated costs would make this unfeasible. Thus, programming classrooms often use automated assessment systems [30]. Automated assessment systems can enhance the efficiency and scalability of the assessment process, making it possible to provide immediate feedback to students and ensure a fair and unbiased evaluation as every student's work is subjected to the same criteria. Despite the benefits, automated assessment has limitations as it mainly focuses on assessing the correctness of student-written code, failing to capture and aid in the problem-solving process [15].

The recent rise of large language models has been highlighted as an additional avenue that could improve—and will certainly impact—computing education [1, 3, 7]. Researchers have already explored the capabilities of large language models, highlighting their potential in writing code and solving and creating programming assignments [9, 35, 37], explaining code [19, 24], identifying programming concepts [38], improving programming error messages [20, 36], and responding to students' help requests [10]. While most of the studies on generative AI and LLMs in the context of computing education are based on expert evaluation of model outputs and single-shot experiments, the studies highlight the potential of using LLMs for formative feedback.

To highlight this, studies exploring the use of LLMs as teaching assistants are starting to emerge [17, 18, 32]. While expert evaluations and one-off experiments provide valuable insights, there is a need for further studies that consider the potential and impacts of LLM-based teaching assistants—or chatbots—in computing education. Perhaps the closest match to our work is that of Prasad et al. [32] who found that students did not use LLMs much beyond the assignment where they were introduced in an upper-level course when students were provided free, unrestricted access to LLMs through a visual studio code plugin.

3 METHODOLOGY

3.1 Context and chatbot

The experiments were conducted at courses offered by Aalto University in Finland. The courses in question use an online learning platform that allows hosting interactive ebooks with embedded assignments. During the summer of 2023, we integrating a LLM-based chatbot to the course platform. The chatbot was based on OpenAI APIs and students could engage in dialogue with it. We intentionally did not conduct any prompt engineering to e.g. constrain or modify the responses, and allowed students to use the chatbot similarly to how one would use a dialogue-based system like ChatGPT.

The chatbot is available for learners on each course material page. When clicking an icon indicating the chatbot, the chatbot opens up in a modal window in which it can be conversed with. Students were made aware that any communication that they had with the chatbot would be stored both on the course platform and sent to the OpenAI APIs. The use of the chatbot was limited to 5 messages per minute and to 100 messages per day per user.

The platform and the courses that use the chatbot informed students of the chatbot and the policy associated to using the chatbot, highlighting that the chatbot is an assisting technology and that using the chatbot for creating solutions for assignments is not acceptable. The policy of use was provided as follows.

> In the Fall of 2023, we introduced a large language model -based generative AI assistant to the course platform. You can find it on the lower right corner of the material pages when logged in – clicking it opens up a chat. The current version of the course assistant is based on ChatGPT. The assistant is not a TA, but a tool to help you with the course. Similarly to asking information from your peers, course teachers, and TAs, you can use the AI assistant to help you with the course and the materials. You can, for example, ask for it to provide additional information about a topic, to explain code, to identify bugs in your code, and so on. You can also ask it for help when you are stuck e.g. with a programming assignment.
>
> > There are humans available for help as well, as discussed in the part on "Asking for help and discussion area".
>
> Do not use the assistant for creating solutions to the assignments, or ask it to complete the assignments for you, as this is harmful for learning. Like using solutions from others, using solutions generated by generative AI and large language models constitutes as plagiarism.
>
> > The use of generative AI and large language models such as ChatGPT for completing coursework on your behalf is not allowed. Using solutions from ChatGPT or similar relates to representing the work of others as your own. When submitting coursework, only use solutions constructed by yourself.

> If you are uncertain whether your use of the assistant is allowed, please ask the course staff, and keep in mind that you are responsible for your own learning. A good way to rehearse and assess whether you have internalized the concepts and that you have worked on with your peers, TAs, or the assistant (etc), is to take a 30 minute break after the collaboration and complete (or redo) the problems on your own.

3.2 Courses

During the fall of 2023, the chatbot was in three courses offered using the platform: Software Engineering with Large Language Models (3 ECTS credits); Device-Agnostic Design (5 ECTS credits); and Web Software Development (5 ECTS credits). The Software Engineering with Large Language Models (SE with LLMs) course was a tailored course for software engineers working in the industry. The course introduces principles of LLMs, including how they work and how they are prompted, and broadly discusses leveraging them in different phases of the software development life cycle. Students completed tasks from the software development life cycle, including documentation tasks, programming tasks, and testing tasks.

The Device-Agnostic Design (DAD) course is a first-year MSc course that focuses on the principles of designing applications that work on a wide range of devices with multiple possibilities for input modalities. The course projects used Dart and Flutter as the technologies. Finally, the Web Software Development (WSD) course is a 2nd year Bachelor's level computer science course where students learn to design and implement web applications. In the course, students used Deno and Hono for building server-side functionality. Notably, the course tries to leverage new technologies and introduced also Deno KV[2] and used Svelte and SvelteKit[3] for building the client-side functionality.

One of the authors of this article is the responsible teacher of all of the three courses.

3.3 Surveys and feedback

The course platform had a brief background survey and a brief feedback item for providing feedback on the utility of the LLM based chatbot. Providing survey answers and feedback was voluntary and students were not compensated for answering in any way.

3.3.1 Background survey. The background survey asked for experience in programming and in the use of LLMs. The questionnaire contained three items, which were as follows.

(1) On a range from 'Not at all experienced' to 'Very experienced', how would you characterize your prior programming experience?
(2) If you have written programs before, in lines of code, what is the largest program you have written? (NA=not applicable)
(3) On a range from 'Not at all experienced' to 'Very experienced', how would you characterize your experience of using large language models (e.g. ChatGPT, GitHub Copilot, ...)?

Items 1 and 3 were answered using a scale from 1 to 9, where 1 corresponded to not at all experienced, while 9 corresponded to very experienced. Item 2 was responded to using the following options: NA, Under 500, 500-5000, 5001-40000, and over 40000.

3.3.2 Usefulness of the chatbot. At the end of every dialogue with the chatbot, the course material opened a dialog with the question "How useful was the chatbot?" that could be answered with a rating ranging from 1 star to 5 stars.

3.4 Data collection and filtering

All data was collected during the Fall of 2023 and processed accordingly to the national ethical guidelines. No ethical review was required. Overall, during the Fall of 2023, 257 students used the chatbot. From these, 228 provided research consent (89%, which is similar to the overall research consent rate in the courses). From the 228, 14 did not actively participate in any of the courses and were omitted from the analysis. Thus, the analyses on chatbot usage focus on 214 students. As the background survey was optional, not all 214 students provided background data.

3.5 Usage coefficient analysis

In order to study whether there are differences between courses and course chapters in how students used the chatbot, we calculated a usage coefficient for each course chapter. First, we calculated the average chatbot use for each student separately. Then, for each student, we calculated a coefficient comparing their chatbot use in each chapter to their personal average. For example, a coefficient of 0.5 would indicate that the student used the chatbot only half of their usual average, while a coefficient of 2 would mean they used the chatbot twice as much as their average for a specific chapter. Then, for each chapter, we calculated the average coefficient over all students. This allows calculating statistics such as the ranges and the standard deviations of the coefficients per course, both of which can indicate the magnitude of differences between chapters. The results of this analysis are presented in Section 4.3.

4 RESULTS

4.1 Descriptive statistics

Descriptive statistics of the chatbot usage per course are outlined in Table 1.[4] The usage of the chatbot was highest in the SE with LLMs course, where 98% of the participants used the chatbot. In the two other courses, the usage was lower, where 22% and 24% of the participants used the chatbot for the DAD course and the WSD course, respectively. As shown in the table, the amount of messages also differed considerably between the courses, ranging from an average of 4 messages per chatbot user (in DAD) to an average 100 messages per chatbot user in SE with LLMs.

Table 2 outlines statistics from participants' self-reported prior experience collected using the survey outlined in 3.3.1. Overall, participants in SE with LLMs rated their programming experience as somewhat higher than those in the other courses. On the other hand, participants in the other courses self-reported their experience with LLMs as somewhat higher than participants in SE with LLMs.

[2] A globally replicated low-latency key-value database announced in May 2023.
[3] The course used Svelte 5 alpha, which was released in November 2023, one week before the first lecture that focused on building client-side functionality.
[4] Note that as a few students took part in multiple courses, the sum of the individual courses' participant counts is slightly larger than the total number of students overall.

Table 1: Descriptive statistics of chatbot usage per course. Users are those students who used the chatbot, while course participants is the number of students in the courses. Only students with research consent are reported.

Course	Users / Course participants	Messages
(1) SE with LLMs	59 / 60 (98%)	5916
(2) DAD	24 / 109 (22%)	99
(3) WSD	135 / 554 (24%)	1094

Table 2: Participants' self-reported prior overall programming experience (Prog. Exp.), programming experience in lines of code (LOC), and experience of using large language models (LLM exp.). Symbol μ denotes mean and symbol η denotes median.

Course	n	Prog. Exp.		LOC		LLM Exp.	
		μ	η	μ	η	μ	η
(1) SE with LLMs	47	6.1	7	2.4	2	2.7	2
(2) DAD	8	5.1	6	1.8	2	4.5	4
(3) WSD	73	5.0	5	1.9	2	4.2	4

Table 3: Chatbot Usage Coefficients per Course

Course	Mean	Median	SD	Range
(1) SE with LLMs	0.91	0.97	0.45	[0.26, 1.77]
(2) DAD	1.00	0.97	0.23	[0.72, 1.32]
(3) WSD	0.98	0.96	0.22	[0.64, 1.45]

4.2 Chatbot usage per course

Chatbot usage was highly variable between the students, producing a distribution that, at least visually, appears roughly Zipfian. While the two most active users had over 400 messages with the chatbot, the third most active student had under 300 messages. Of the 214 students who used the chatbot, 18 students produced over one-half of the messages. Only 61 (28.5 %) had 25 or more messages, while 85 students (39.7 %) had 10 or more messages.

Observing the courses in isolation, we note that both the SE with LLMs course, and the WSD course, exhibit the same phenomenon. In both courses, two power users have significantly more interactions with the chatbot than others students, with the number of interactions quickly decreasing. These top power users have 423 and 414 messages in the case of SE with LLMs, and 156 and 122 interactions in the case of WSD. For comparison, the third-most active users on these courses have 262 and 49 interactions, respectively. These power users are distinct students. For the Device Agnostic Design course, the usage levels are generally very low, with the highest number of messages for any student being 12. The per-course usage distributions are shown in Figure 1.

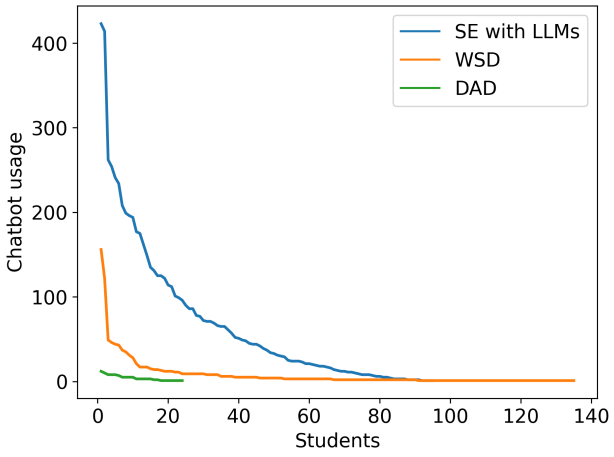

Figure 1: Student chatbot usage distribution per course.

4.3 Chatbot usage per chapter

We further looked into the use of the chatbot in individual chapters of the material, focusing on deviations from average usage behavior. The aggregate statistics for the three courses are shown in Table 3. From the table, we can see that usage was quite similar between the WSD course and the DAD course, but the SE with LLMs course showed different usage behavior. In the SE with LLMs course, the standard deviation and the range of coefficients was larger, suggesting that there were larger differences between individual chapters of the material in how much students used the chatbot.

Overall, in the SE with LLMs, the chatbot was most used in a chapter on tooling and working with Python, which included a range of programming problems, explicitly allowing students to use the chatbot for solving them to demonstrate LLM code generation capabilities. The second chapter with the most use was a chapter that focused on building a larger application with the help of LLMs, starting from decomposing the problem and resulting with an application with a graphical user interface. The chapters with the least chatbot usage focused on review and testing and software engineering, neither of which had assignments and both of which discussed the topics on a higher level.

In the DAD course, the least usage was observed in a chapter on Flutter basics, which introduced participants to showing simple content in a Flutter application, while the most usage was observed in a chapter on handling input with Flutter. Both chapters included programming problems, but the problems in the chapter on handling input were considerably more complex

In the WSD course, the chatbot was most used in a chapter introducing the concept of storing data on server using Deno KV and a chapter on state management with Svelte. Notably, in both of these chapters, the usefulness feedback median was 1, indicating that the chatbot was not at all useful – very likely due to the technologies being so new that the LLM would suggest deprecated approaches. The chatbot was least used in a chapter on data validation, which introduced the principles of validating data, and introduced a library for the task.

Table 4: Average usefulness of the chatbot in each of the courses. The median usefulness was 4 (out of 5) in all courses.

Course	Ratings	Average Usefulness
(1) SE with LLMs	456	3.8
(2) DAD	35	3.1
(3) WSD	242	3.1

Table 5: Correlation coefficients between chatbot usage and student experience variables. Data contains only those students who responded to the experience survey. Bolded values have $p < 0.001$.

	Usage	Prog. Exp.	LOC	LLM Exp.
Usage	**1.00**	0.10	0.11	**-0.41**
Prog. Exp.		**1.00**	**0.58**	0.16
LOC			**1.00**	0.19
LLM Exp.				**1.00**

4.4 Chatbot usefulness

At the end of every dialogue, students were prompted to rate the usefulness of the chatbot using a rating from 1 to 5 stars. Table 4 outlines the results. Overall, students in the SE with LLMs considered the chatbot as somewhat more useful (avg. rating 3.8/5), than students in the other courses who rated the chatbot on average 3.1/5. The median usefulness in all of the courses was 4 out of 5. These numbers, however, need to be considered in the context of a possible self-selection bias: a student who trials the chatbot once and determines it unusable would produce only a single low rating, while superusers happy with the chatbot might produce hundreds of high ratings. We return to this topic in the discussion.

4.5 Usage and student backgrounds

Using Spearman's Rho (Table 5), we also observed that chatbot usage, measured as the number of messages, was moderately negatively correlated with students' previous experience with LLMs ($\rho = -0.41, p < 0.01$). We hypothesize that this decrease in usage with experience can be explained by inexperienced students running various tests and trials to get a better feel of the LLM, which those already familiar with LLMs would presumably not conduct at least to the same degree. On the other hand, as we briefly discuss in our study limitations, students with more experience with LLMs may have access to LLMs through other means. The correlations between chatbot usage and prior programming experience or largest program written were not statistically significant ($\rho = 0.10$, $p = 0.25$ and $\rho = 0.11$, $p = 0.20$, respectively).

5 DISCUSSION

5.1 Course and population differences

The courses differed in terms of participants and chatbot usage. The SE with LLMs was attended by software engineers from the industry who rated their prior programming experience higher than students in the other courses. At the same time, students rated their prior experience with LLMs higher than the software engineers. The differences in programming experience was to be expected, while we were somewhat surprised with the difference in LLM experience. Students may be more open to rapidly adapting new technologies when contrasted to professional software engineers – software engineers might be constrained in terms of the tools that they can adapt, and larger companies can still be vary of LLMs due to existing legal disputes[5] and uncertainties. In our experience, students are actively discussing LLMs, and we have observed a significant increase, e.g., in the quantity of theses related to LLMs.

5.2 Chatbot usefulness

Our results indicate that at least *some* students find LLMs highly useful, becoming powerusers, but at the same time a significant amount of students barely engage with them. As the usefulness ratings collected from the students are dominated by the first group, they should be interpreted with caution: while the intrinsic component of the evaluation was positive, it suffers from a potential self-selection bias and the main proxy for extrinsic effectiveness, actual usage, offers a less clear view. Further study is clearly needed.

At the same time, these results on the use and usefulness of LLMs were also to be expected. While the SE with LLMs course explicitly instructed participants to use LLMs, the other courses offered the chatbot more as an additional support mechanism. This already can impact the use of the chatbot significantly. In our case, less than 10% of the students who used the chatbot produced more than 50% of the messages. This also aligns with prior work that found that most students did not use an LLM-based chatbot beyond initially trying it out when it was introduced [32].

When considering the relative differences in how students used the chatbot in the courses, we see parallels to the use of help resources in online courses. Students differ in how, when, and from whom they ask for help [27]; in online courses, the majority of participants do not engage in discussions, while some are very active, even to be labeled as "superposters" [13, 27]. As prior research has highlighted that there are students who perhaps read posts but do not necessarily comment on posts or ask questions [16], a possible future stream of research would be to identify and highlight discussions with the chatbot that have been very useful for learning, and allow sharing them to other course participants on the online platform.

Similarly, the average usefulness differed between the courses. The higher usefulness of the chatbot likely relates to the direct use for course tasks, while the lower usefulness in other courses could relate to course technologies. Both DAD and WSD have students work on larger projects with multiple files, which might not be very convenient with the chatbot. In addition, both courses also keep up to date with technology versions; as an example, WSD used Svelte 5 alpha as the frontend technology, which was released just before the start of the classes that focused on building frontend functionality. As LLMs have a knowledge cutoff point that reflects the time when the data that was used for training, LLMs in general do not have information of technologies released after specific moments in time. One potential direction for future work would be to add retrieval augmented generation functionality to the course materials, which

[5] See e.g. https://githubcopilotlitigation.com/

would allow the LLMs to retrieve information from the materials when creating a response.

5.3 Instructor viewpoint

Overall, when considering the possibility of embedding an LLM-based chatbot to the course platform, we noted that students are already gaining experience from using LLMs and some students also have accounts to services such as OpenAI ChatGPT. By providing access to a chatbot that leverages a state-of-the-art LLM, we created a possibility of leveling the playing field, where students could use a state-of-the-art LLM even if they would not be paying for it. The cost of using the OpenAI API was less than $200 for the whole fall semester.

While the SE with LLMs was a new course, DAD and WSD are courses that have been offered in previous years. The courses allow students to ask for help through the online platform and offer labs for students where they can ask for help. The introduction of the LLM-based chatbot did not affect the use of the help request functionality or the labs in an observable manner. Prior research has pointed out that making sample solutions available to students can lead to reduced use of support [29]; although the study contexts are different, we can speculate that the LLM-based chatbot was not simply seen as a source for sample solutions.

The courses also use tools for plagiarism detection that use fingerprinting and compare solutions to detect similarities. In a surface-level post-hoc analysis, we did not observe noticeable differences in plagiarism, nor did we identify students explicitly seeking to simply use the models for solving their assignments. We however acknowledge that detecting LLM-generated content can be difficult, and the courses may have suffered from the problem already earlier.

Perhaps one of the key observations from class discussions was that the chatbot was rather poor at helping with errors and with debugging larger code, which was also included in the few written feedbacks. While earlier research has highlighted the possibility of using LLMs for enhancing programming error messages [20, 36], the prior studies have been conducted with introductory-level programming assignments that are relatively small and well-scoped. In our context, the applications where students needed help were typically larger, consisting of multiple files. This highlights also the need to consider moving towards IDE-integrations in chatbots, as has already been done with e.g. GitHub Copilot.

5.4 Limitations of work

This study comes with a range of limitations, which we discuss here. First, we acknowledge that students may have used LLMs also through other means, for example through their own accounts on LLM providers. While we did not ask whether students used other LLMs, it is possible that this could be the case. Second, responding to the various survey instruments was voluntary. While we received over 700 responses to the brief usefulness rating that students could answer by clicking once, as the feedback was tied to actually using the chatbot, the responses naturally underrepresent students who did not extensively—or at all—use the chatbot. Third, as the student background was self-reported, it might not accurately reflect the real ground-truth experience levels of the students. Fourth, while we built the chatbot on the OpenAI APIs, we did not collect information on how much time creating the response took. Especially on larger responses, the time to produce the response can be considerable, which by itself can already reduce the perceived usefulness of the chatbot.

We also acknowledge that OpenAI APIs were under a denial of service attack on a few days, which could also influence the time to form a response – or even whether a response was formed at all. We also note that the chatbot was embedded into the learning materials, and not e.g. to any programming environment that the students used. This likely influences the usefulness of the chatbot especially with larger assignments. Finally, this type of "intrinsic" evaluation of a natural language generation system is generally viewed as inferior to an "extrinsic" evaluation focused on how the system allows the user to complete some specific task [4, 40]. Our analysis also focuses on the *average* performance of the chatbot, an approach that ignores the potential for catastrophic errors that—even if rare—could meaningfully affect whether a chatbot like this is in reality an ethically feasible learning aid [28].

6 CONCLUSION

In this work, we report our experiences from giving students access to an LLM-based chatbot in 2023. We purposefully made the chatbot similar to commercially available chatbots, so that student use would hopefully be as authentic as possible. Our experiences suggest that the worst fears of educators—most students developing severe overreliance on LLM chatbots—might not materialize, even if the chatbot does not have any "guardrails" [21].

To summarize, our answers to the research questions are as follows: (RQ1) The use of the chatbot differed considerably between the courses and to some extent between course material chapters, where the chatbot was most used in a course that taught participants to use LLMs in software engineering. The other two courses saw less use of the chatbot, and there was less variation in the amount of use between course materiel chapters. (RQ2) The chatbot was perceived as most useful in the course that focused on LLMs in software engineering (average 3.8/5), while participants in other courses rated the interaction as somewhat less useful (average 3.1/5). In all courses, the feedback median for usefulness was 4. Anecdotal evidence highlighted that the students did not find the chatbot very useful for debugging or improving error messages, despite some previous works highlighting LLMs' potential for these tasks [20, 36]. Similarly, the usefulness of the chatbot in course material chapters that involved very recent technologies was low. (RQ3) Previous experience with LLMs was linked with lower use of the chatbot, but prior programming experience was not related to the chatbot usage.

As a part of our future work, we are collecting data from additional courses and looking in more detail into the role of the course in relationship with chatbot usage. In addition, we are exploring the relationship of self-regulation, large language model use, and perceptions and conceptions of plagiarism.

ACKNOWLEDGMENTS

This research was supported by the Research Council of Finland (Academy Research Fellow grant number 356114).

REFERENCES

[1] Brett A. Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, Lauri Malmi, James Prather, and Keith Quille. 2023. Generative AI in Introductory Programming Education. https://csed.acm.org/large-language-models-in-introductory-programming/ CS2023 Curricular Practices Volume.

[2] Benjamin S Bloom. 1984. The 2 sigma problem: The search for methods of group instruction as effective as one-to-one tutoring. *Educational researcher* 13, 6 (1984).

[3] Peter Brusilovsky, Barbara J. Ericson, Cay S. Horstmann, Christian Servin, Frank Vahid, and Craig . 2023. Significant Trends in CS Educational Material: Current and Future. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 2*. ACM, NY, NY, USA, 1253. https://doi.org/10.1145/3545947.3573353

[4] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *arXiv preprint arXiv:2006.14799* (2020).

[5] Geert De Clercq and Josie Kao. 2023. Top French university bans use of ChatGPT to prevent plagiarism. *Reuters* (2023). https://www.reuters.com/technology/top-french-university-bans-use-chatgpt-prevent-plagiarism-2023-01-27/

[6] Paul Denny, Brett A. Becker, Juho Leinonen, and James Prather. 2023. Chat Overflow: Artificially Intelligent Models for Computing Education - RenAIssance or ApocAIypse?. In *Proc. of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ACM, NY, NY, USA, 3–4.

[7] Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. 2024. Computing Education in the Era of Generative AI. *Commun. ACM* (2024). https://doi.org/10.1145/3624720

[8] Benedict Du Boulay. 1986. Some difficulties of learning to program. *Journal of Educational Computing Research* 2, 1 (1986), 57–73.

[9] 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*. ACM, NY, NY, USA, 10–19.

[10] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutcheme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proc. of the 2023 ACM Conference on International Computing Education Research - Volume 1*. ACM, NY, NY, USA, 93–105.

[11] Hour of Code. [n. d.]. Blurps and useful stats. https://hourofcode.com/us/promote/stats. Accessed: 2024-01-05.

[12] White House. 2016. Fact sheet: President obama announces computer science for all initiative. *Retrieved [2024-01-04] from https://obamawhitehouse.archives.gov/the-pressoffice/2016/01/30/factsheet-president-obama-announces-computer-science-all-initiative* (2016).

[13] Jonathan Huang, Anirban Dasgupta, Arpita Ghosh, Jane Manning, and Marc Sanders. 2014. Superposter behavior in MOOC forums. In *Proceedings of the first ACM conference on Learning@ scale conference*. 117–126.

[14] Majeed Kazemitabaar, Justin Chow, Carl Ka To Ma, Barbara J. Ericson, David Weintrop, and Tovi Grossman. 2023. Studying the Effect of AI Code Generators on Supporting Novice Learners in Introductory Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*. ACM, NY, NY, USA, Article 455, 23 pages. https://doi.org/10.1145/3544548.3580919

[15] Hieke Keuning, Johan Jeuring, and Bastiaan Heeren. 2018. A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education (TOCE)* 19, 1 (2018), 1–43.

[16] René F Kizilcec, Mar Pérez-Sanagustín, and Jorge J Maldonado. 2017. Self-regulated learning strategies predict learner behavior and goal attainment in Massive Open Online Courses. *Computers & education* 104 (2017), 18–33.

[17] Harsh Kumar, Ilya Musabirov, Mohi Reza, Jiakai Shi, Anastasia Kuzminykh, Joseph Jay Williams, and Michael Liut. 2023. Impact of Guidance and Interaction Strategies for LLM Use on Learner Performance and Perception. *arXiv preprint arXiv:2310.13712* (2023).

[18] Harsh Kumar, Ilya Musabirov, Joseph Jay Williams, and Michael Liut. 2023. QuickTA: Exploring the Design Space of Using Large Language Models to Provide Support to Students. In *Workshop on Partnerships for Co-Creating Educational Content at the Learning Analytics and Knowledge Conference*.

[19] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. In *Proc. of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. ACM, 124–130.

[20] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A. Becker. 2023. Using Large Language Models to Enhance Programming Error Messages. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, NY, NY, USA, 563–569.

[21] Mark Liffiton, Brad E Sheese, Jaromir Savelka, and Paul Denny. 2024. CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes. In *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research*. ACM, NY, NY, USA, Article 8, 11 pages.

[22] Andrew Luxton-Reilly, Ibrahim Albluwi, Brett A Becker, Michail Giannakos, Amruth N Kumar, Linda Ott, James Paterson, Michael James Scott, Judy Sheard, and Claudia Szabo. 2018. Introductory programming: a systematic literature review. In *Proc. Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education*. 55–106.

[23] Stephen MacNeil, Joanne Kim, Juho Leinonen, Paul Denny, Seth Bernstein, Brett A Becker, Michel Wermelinger, Arto Hellas, Andrew Tran, Sami Sarsa, et al. 2022. The Implications of Large Language Models for CS Teachers and Students. In *Proc. of the 54th ACM Technical Symposium on Computer Science Education V. 2*.

[24] 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. of the 54th ACM Technical Symposium on Computer Science Education V. 1*. ACM, NY, NY, USA, 931–937.

[25] Rohan Mehta. 2023. Banning ChatGPT will do more harm than good. *MIT Technology Review* (2023). https://www.technologyreview.com/2023/04/14/1071194/chatgpt-ai-high-school-education-first-person/

[26] Diba Mirza, Phillip T Conrad, Christian Lloyd, Ziad Matni, and Arthur Gatin. 2019. Undergraduate teaching assistants in computer science: a systematic literature review. In *Proc. of the 2019 ACM Conf. on Int. Computing Education Research*.

[27] Matti Nelimarkka and Arto Hellas. 2018. Social help-seeking strategies in a programming MOOC. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 116–121.

[28] Jekaterina Novikova, Ondřej Dušek, Amanda Cercas Curry, and Verena Rieser. 2017. Why We Need New Evaluation Metrics for NLG. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Copenhagen, Denmark, 2241–2252.

[29] Henrik Nygren, Juho Leinonen, and Arto Hellas. 2019. Non-restricted Access to Model Solutions: A Good Idea?. In *Proc. of the 2019 ACM Conf. on Innovation and Technology in Computer Science Education*. 44–50.

[30] José Carlos Paiva, José Paulo Leal, and Álvaro Figueira. 2022. Automated assessment in computer science education: A state-of-the-art review. *ACM Transactions on Computing Education (TOCE)* 22, 3 (2022), 1–40.

[31] Neil Perry, Megha Srivastava, Deepak Kumar, and Dan Boneh. 2023. Do users write more insecure code with AI assistants?. In *Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security*. 2785–2799.

[32] Siddhartha Prasad, Ben Greenman, Tim Nelson, and Shriram Krishnamurthi. 2023. Generating Programs Trivially: Student Use of Large Language Models. In *Proceedings of the ACM Conference on Global Computing Education Vol 1*. 126–132.

[33] 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 Petersen, Raymond Pettit, Brent N. Reeves, and Jaromir Savelka. 2023. The Robots Are Here: Navigating the Generative AI Revolution in Computing Education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. ACM, New York, NY, USA, 108–159. https://doi.org/10.1145/3623762.3633499

[34] James Prather, Brent N. Reeves, Paul Denny, Brett A. Becker, Juho Leinonen, Andrew Luxton-Reilly, Garrett Powell, James Finnie-Ansley, and Eddie Antonio Santos. 2023. "It's Weird That It Knows What I Want": Usability and Interactions with Copilot for Novice Programmers. *ACM Trans. Comput.-Hum. Interact.* 31, 1, Article 4 (nov 2023), 31 pages. https://doi.org/10.1145/3617367

[35] Ben Puryear and Gina Sprint. 2022. Github copilot in the classroom: learning to code with AI assistance. *J. of Computing Sciences in Colleges* 38, 1 (2022), 37–47.

[36] Eddie Antonio Santos, Prajish Prasad, and Brett A Becker. 2023. Always Provide Context: The Effects of Code Context on Programming Error Message Enhancement. In *Proc. of the ACM Conference on Global Computing Education Vol 1*. 147–153.

[37] 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, NY, USA, 27–43. https://doi.org/10.1145/3501385.3543957

[38] Andrew Tran, Linxuan Li, Egi Rama, Kenneth Angelikas, and Stephen MacNeil. 2023. Using Large Language Models to Automatically Identify Programming Concepts in Code Snippets. In *Proc. of the 2023 ACM Conf. on Int. Computing Education Research - Volume 2*, Vol. 1. ACM, 563–569.

[39] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models. In *CHI Conference on Human Factors in Computing Systems Extended Abstracts*. Association for Computing Machinery, NY NY, USA, 1–7.

[40] Chris van der Lee, Albert Gatt, Emiel van Miltenburg, and Emiel Krahmer. 2021. Human evaluation of automatically generated text: Current trends and best practice guidelines. *Computer Speech & Language* 67 (2021), 101151.

[41] Arto Vihavainen, Jonne Airaksinen, and Christopher Watson. 2014. A systematic review of approaches for teaching introductory programming and their influence on success. In *Proc. of the tenth annual conf. on Int. computing education research*. 19–26.

[42] Christopher Watson and Frederick WB Li. 2014. Failure rates in introductory programming revisited. In *Proc. of the 2014 conference on Innovation & technology in computer science education*. 39–44.