A Preliminary Analysis of Students' Help Requests with an LLM-powered Chatbot when Completing CS1 Assignments

Ruiwei Xiao[1], Xinying Hou[2], Harsh Kumar[3], Steven Moore[1], John Stamper[1] and Michael Liut[4]

[1]Carnegie Mellon University
[2]University of Michigan
[3]University of Toronto
[4]University of Toronto Mississauga

Abstract

Multiple recent studies have integrated large language models (LLMs) into diverse educational contexts, including CS1 classrooms. One common application is integrating a chatbot to serve as a teaching assistant. In this preliminary analysis, we explored four methods (correlation analysis, Latent Dirichlet Allocation, expert evaluation, LLM labeling, and evaluation) with multiple levels of data to analyze students' help requests with a basic chat-based LLM tutor when completing CS1 assignments. This dataset contains 73 initial help-seeking conversation sessions with corresponding student self-reported survey answers. It also included 18 hallucinating responses from all the conversation sessions. Our results indicate that students with lower self-efficacy tended to create longer help requests, while students with higher self-efficacy tended to conduct more concise ones. Other than this, we found that learners shared more commonalities than differences when conducting help requests, including the length of turn-taking and the struggle to locate LLM hallucinations. As AI-based chatbots become prevalent in education settings, this preliminary analysis sheds light on what types of learner data can be collected, and what analytic approaches can be leveraged to unpack students' help-seeking with these LLM-based learning systems.

Keywords

large language models, computing education, intelligent tutors, learner-centered design, field experiment

1. Introduction

The rise of large language models in educational contexts has marked a significant increase in the use of LLMs for teaching and learning [1, 2]. These models have been integrated into various educational settings, including introductory computer science (CS1) education, providing innovative ways to support student learning [3, 4, 5]. As the use of LLM-powered learning systems became prevalent, taking a closer look at students' educational behaviors with these systems became important. Previous educational data mining work highlights that there are different types of features in the datasets, such as demographic features, performance features, and activity/engagement features [6]. In this work, we collected a comprehensive set of data and conducted a preliminary analysis of how students interact with a chat-based LLM tutor to complete a CS1 assignment. Through a mixed-method analysis of multifactual data, our study reveals that, while self-efficacy is negatively correlated with the length of a student's initial help request, learners exhibit few differences in other help-seeking behaviors. Instead, students share common flaws in their help-seeking strategies, such as providing insufficient information in the help request, or failing to locate errors in the LLM hallucination. These findings highlight some needs for future research and development of LLM-based educational systems. For instance, these systems should not only be more learner-centered and adaptive, but also capable of automatically recognizing and scaffolding the context of learners' questions, thereby supporting students from diverse backgrounds more effectively.

By using multiple methods and data levels to understand student help-seeking with a chat-based LLM tutor, this work aims to contribute to the long line of research on designing educational tools that are sensitive to learners' individual needs and features. By addressing the identified commonalities and differences in help-seeking behaviors, we aimed to inform the development of a more effective, context-aware LLM-based educational system that can enhance the learning experience for all students, regardless of their backgrounds.

2. Related Work

2.1. LLM-based Programming Tutors

Prior studies showed that LLM can generate code, explanations, and conversations [7]. After uncovering its potential, significant effort has been invested in creating and evaluating LLM-based intelligent programming tutors with diverse content focus and granularity. However, as LLM products have shown enhanced flexibility and are more accessible to students, there are rising concerns about their over-utilization in computing education [3]. Therefore, recent developments have emphasized incorporating LLM with intelligent programming tutors in a more pedagogical way [8]. One direction is to provide an LLM-powered middle-stage code puzzle to support students actively in completing their programming experience [9]. The other direction is to provide help similar to that of a teaching assistant to support students in completing programming exercises but avoid providing direct code [10, 11]. The QA bot from which we collected data in this paper is from this direction. In this work, we are mainly interested in understanding how students interact with the QA bot and how their help-seeking queries relate to their learning backgrounds, such as self-efficacy and fluency in English.

CSEDM'24: 8th Educational Data Mining in Computer Science Education Workshop, July 14, 2024, Atlanta, GA
ruiweix@andrew.cmu.edu (R. Xiao); xyhou@umich.edu (X. Hou); harsh@cs.toronto.edu (H. Kumar); stevenmo@andrew.cmu.edu (S. Moore); jstamper@cmu.edu (J. Stamper); michael.liut@utoronto.ca (M. Liut)

2.2. Granularity of Data in Current LLM-Based Tutoring Data Analysis

The multilevel nature of data in social sciences is pervasive, however, reporting practices for data aggregation lack standardization, resulting in considerable variability in the information and statistics included by authors [12]. In the learning engineering domain, the levels of data have been categorized based on their granularity from low to high as: ***log data*** (records of students' usage events, e.g. xAPI data [13]), ***inferred data*** (characteristics or prediction of behaviors of the data subjects inferred from other data, e.g. anxiety level [14], student model [15]), ***aggregate data*** (the aggregation of multiple events or student's data, e.g. learning curve analysis [16]), ***program-level data*** (course or program-level data used to track or predict the performance of a group of students during the entire course or program, e.g. course recommendation data [17]), and ***profile data*** (student's personal information, such as demographic data [18], learning preferences, behavioral data, goals and aspirations, socio-emotional data, etc.) [19].

To analyze the usage of LLM-based programming tutors, most of the existing works conducted analysis using log data, inferred data from surveys, and aggregate data of LLM-tutor usage; few of them have analysis on profile data on personalized learning experiences and program-level inferences. To bridge the gap, we use our system, QuickTA [20], to collect the data levels mentioned above, and explore extensive approaches and aspects of analysis on help-seeking requests with emphasis on profile data to understand to what extent learners' differences would influence the usage of LLM-tutors.

2.3. Self-Efficacy and Help-Seeking

Self-efficacy refers to individuals' subjective evaluations of their abilities to successfully perform an activity [21]. In computer science education, students' self-efficacy refers to their perception of competence to complete CS courses and finish programming tasks [22, 23]. Prior research looked into how specific learning supports might impact student self-efficacy in CS learning [24] and whether the advantages of support mechanisms could benefit students with varying levels of self-efficacy [22, 25]. According to the relationship between students' self-efficacy and help-seeking behaviors, results from previous work are mixed [26]. Some scholars found that students with high self-efficacy tend to show high help-seeking behavior [27, 28]. However, others reported the opposite direction and found that students with a high sense of self-efficacy avoid seeking help even in times of need [29].

As such, the student's self-efficacy might affect their use and interaction with an LLM tutor. We collected this type of data by asking students to self-report their self-efficacy regarding the assignment topic each time before interacting with the LLM tutor.

2.4. Impact of English Fluency on Help-Seeking

Previous research has examined user behavior in searching for information in English as a foreign language [30]. When using English to search, non-native speakers identified the query formulation as the most challenging task. Due to their relatively low English language proficiency level, identifying keywords to build a query in a non-native language is their main difficulty [30]. Similarly to online search, an important step in using an LLM-based QA bot is also formulating an appropriate query [31]. However, limited work has been done to investigate the impact of English fluency on Learner-LLM interactions in the context of a CS course.

Given that our context involved a significant number of non-English native speakers, this provided us a chance to investigate the relationship between students' self-reported level of English fluency and their interactions with the LLM tutors.

3. Methods

We deployed QuickTA, an LLM-powered chat-based tutoring system, in a large introduction to computer programming course. Students were given access to QuickTA when solving weekly lab assignments. This section describes the context of the deployment, the design of QuickTA, and the dataset we used in this analysis.

3.1. Classroom Context and QuickTA

This study was conducted within an "Introduction to Computer Programming" course (CS1), offered at a prominent research-intensive post-secondary institution in Canada during the Fall 2023 semester. The course is interdisciplinary, with the majority of students (typically over 75%) in their first year of post-secondary study intending to major in computer science. The 12-week course utilized a flipped classroom model and included ten assignments (starting in week 2 and ending in week 11), pre-and post-homework due weekly, two-term tests (in weeks 6 and 11, respectively), and a final exam held two weeks after the conclusion of the regular semester.

The specific lab assignment we selected for this analysis focused on the topics of *if* statements and *for* loops. At the end of the assignment introduction, students were given a link to access QuickTA Figure 1. Students were informed that QuickTA was designed to help them with the assignment and that they could access it as often as needed. The students were also told that this was an experimental tool, so they should be careful about relying on its responses and should proactively report any issues.

We used GPT-4 (the most advanced model back in Fall 2023) to power QuickTA in helping students with the assignment. As shown in previous research, a system-prompted LLM might be more effective as a tutor [32]. Therefore, we designed a system prompt based on Hattie's feedback model [33]. The prompt was tested internally through multiple iterations with the teaching team before being used for deployment. The detailed configuration of the model is described in Appendix A.

3.2. Dataset

At the time of the assignment, there were 1,068 students enrolled in the class. While QuickTA is available for every lab activity and exam preparation, the interaction with QuickTA is optional. In this analysis, we selected student data for only one lab activity (Lab 4).

For each lab activity, students were asked to complete three questions every time before starting a QuickTA con-

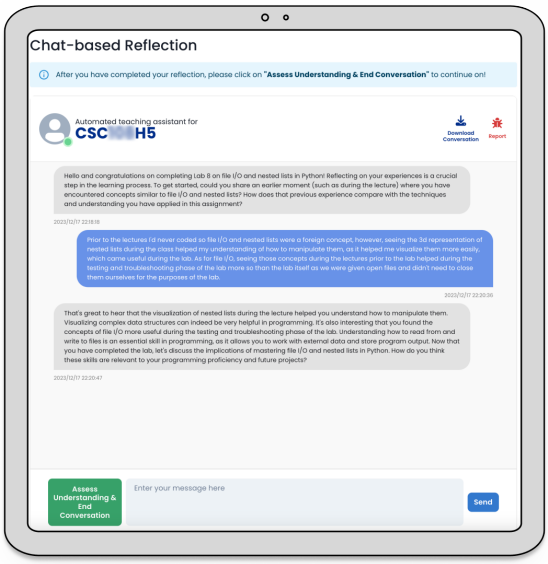

Figure 1: An Example Interface of QuickTA

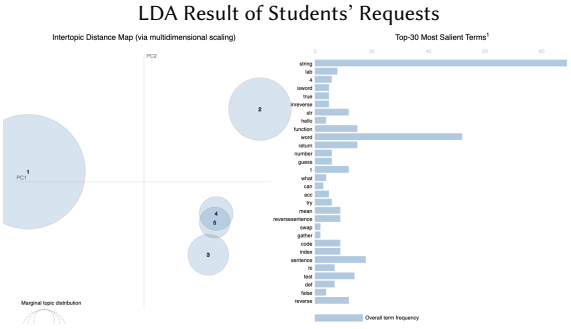

Figure 2: LDA Result of Students' Requests

versation. The definition and average score of each question are as follows: 1) ***self-efficacy*** *(M=4.74, SD=1.38)*: a self-reported question from a scale of 1 to 7, representing the student's confidence on the topic before using QuickTA (1-not confident at all, 7-the most confident); 2) ***English fluency:*** *(M=4.15, SD=0.88)* a self-reported question from a scale of 1 to 7, representing the student's English fluency (1-not fluent at all, 7-the most fluent); and 3) ***conceptual knowledge*** *(M=0.60, SD=0.49)*: a score of a multiple choice question that tests student's conceptual knowledge on this task, indicating how well the student mastered on the topic before using QuickTA.

A total of 73 students used QuickTA when completing lab 4 homework, and completed all the required sections. When using QuickTA, 23 students conducted multiple conversation sessions (closed and then reopened) with QuickTA. As we focused on students' initial help requests with QuickTA, when answering RQ1 and RQ2, we only kept their 73 initial help requests and follow-up turn-takings in these initial conversation sessions. We also noticed that QuickTA sometimes expressed errors when answering students' help requests. Therefore, in RQ3, we looked into how students dealt with these erroneous answers. A total of 18 hallucinating responses emerged from all the conversation sessions.

4. Results

We applied a mixed analysis approach to analyze our multiple levels of data. More specifically, we adopted statistical analysis in 4.1 and 4.2, we applied an LLM model for thematic coding and content extraction unsupervised learning models for topic extraction in 4.2, and qualitative expert evaluation in 4.2 and 4.3.

4.1. RQ1: QuickTA usage with learner feature data

To understand the relationship between learner-level features and students' usage of QuickTA, we investigated the correlations between learner features with the number of turn-takings (one turn is considered as initiated by the student and then QuickTA answered) in each conversation session and the number of words in each initial help request using correlation analyses. We applied Pearson correlation when the two variables were continuous and Point-Biserial Correlation when one of them was dichotomous.

Table 1
Correlation Between Number of Turn-Taking and Learner Features

Variable	r	p-value
Self-Efficacy	.05	.648
English Fluency	.14	.222
Conceptual Knowledge	.12	.314

Table 2
Correlation Between Number of Words in Initial Help Request and Learner Features

Variable	r	p-value
Self-Efficacy	.25	**.029**
English Fluency	.04	.721
Conceptual Knowledge	.06	.618

As a result, we found that students' self-efficacy is negatively correlated to the number of words in each initial help request. No significant differences have been found in any other variables with times of turn-taking and initial help request length.

4.2. RQ2: Information coverage in learners' help requests

A major aspect that influences the length of the help request would be the number of different types of information covered in the content. For instance, the help requests with learner's code or execution messages (normally above 50 words) would be longer than requests with learners' questions only (usually within 20 words). Therefore, we explored the similarities and differences between learners in their queries' information coverage.

We first applied Latent Dirichlet Allocation (LDA) in Figure 2 to explore differences in frequent topics that learners are interested in while finding more commonality than differences in help query content related to learner traits. Without significant differences between most-frequent words in prompts for different learners, students often include words related to lab name (e.g., lab4), programming problem name or description (e.g., ispalindrome, lowercase), and some programming syntax (e.g., return, true).

Table 3
Table of Help Request Types

Help Request Type	Number of Queries	Definition
Debug Output	5	The code can be executed, but the outputs of the test cases are wrong
Debug Syntax	2	The code contains syntax error(s)
Next-Step	30	The student does not know what to do next
Create Testcase	3	The student asks for help in creating new test cases to examine the code
Capability	4	The student is curious about what this bot can do
Conceptual Knowledge	3	The student needs an explanation of the definition of certain concepts
Procedure Knowledge	1	The student asks about how to use a certain structure
Clarification	8	The student asks for clarification on certain questions or homework requirements
Unclear	17	The student only copies the current code or types greeting words such as "hello"

Then we used GPT-4 to label the types of information covered in each help request to further investigate whether certain types of learners are more skillful prompt writers. The information coverage rate can provide another perspective on users' prompting behavior. In this analysis, we focused on whether learners include sufficient context information in their help requests to increase the help-seeking responses' quality of accuracy and concreteness. We acknowledged that "sufficient context" can be different given different request types. Thus we first defined the types of requests adopting from the help-seeking model [34] as requests with clear motivation: debug-output, debug-syntax, next-step, create test cases, knowledge-concept, knowledge-procedure, capability, clarification; and requests with unclear motivation (Table 3). Two domain experts coded learners' initial help request data. These experts identified the potential components that can be included in each help request and labeled what are essential components for different types of requests.

After the definition of request types and information types, we applied a multi-shot prompt engineering strategy with examples to make GPT-4 label each user request using the rubric in Table 4. Lastly, each student's request is grouped into one help-seeking type, and its information coverage is visualized as shown in Figure 3.

Table 4
Table of Information Components Definition

Information Type	Definition *(Yes-1, No-0)*
Question	Does the student ask a specific question to elaborate the purpose of the prompt?
Statement	Does the prompt include the coding problem statement?
Code	Does the prompt include the student's code?
Output	Does the prompt include the output of the code after running the code?
Testcase	Does the prompt include the test case of this problem?

The result of different learners' information coverage indicates no significant difference between learners (Table 5). A common pattern of all learners is that they tend to miss essential context for multiple types of help requests. Overall speaking, only 24.7% of students included all information components identified as important by researchers, with particularly low coverage on queries related to debug output (20%), debug syntax (0%), and next-step (3%). More specifically, when asking for the next step to do, 63.3% of students only include a question (*e.g., "how do I do the function reverse sentence"*) without mentioning their current code or problem description.

Table 5
Correlation Between Information Coverage and Learner Features

Variable	r	p-value
Self-Efficacy	.07	.365
English Fluency	.05	.540
Conceptual Knowledge	<.01	.963

4.3. RQ3: Learner differences when reacting to LLM hallucinations

The previous section reveals the improvement needed in learners' help-initiating behaviors, and in this section, we closely looked at learners' reactions to hallucinating QuickTA-generated responses.

We manually coded all the conversation sessions and identified 18 hallucinating responses in 8 conversation sessions from 6 unique students. Of these 8 conversation sessions, 5 were in the initial conversation session, and 3 were in later conversation sessions. Due to the nature of programming tasks, students can run the code with tests to validate the responses. Therefore, in most cases(94%), students are able to identify the answer that is wrong from the test result. However, only one student successfully debugged with the system and reached the correct code state, while other students failed to identify the source of errors before quitting the conversation. According to the conversation log, the successful student already completed the lab assignment and only wanted to test the capability of our system, thus this student can accurately identify the cause of error in LLMś incorrect responses within one round of turn-taking (*e.g., "Why would sorted_string be initialized with the first character of the string? We would not be sure of its position until during the sorting process rather than prior"*). Other students who had no access to the correct answer are often trapped in the same hallucinating answer for an average of 15 rounds of turn-taking with the problem unsolved, which can potentially cause low learning efficiency and frustration.

5. Discussion

In this work, we applied multiple analysis methods with multifactual data to understand how learners used a chat-based LLM tutor when completing a CS1 lab assignment. In general, we found more commonalities than differences among

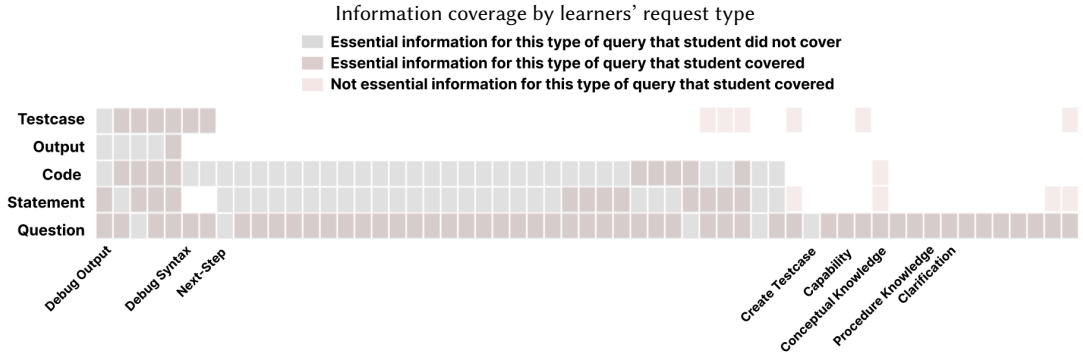

Figure 3: Information coverage by learners' request type. Red: the information students included; gray: essential information to provide to LLMs for higher response quality

different learners regarding their usage, information coverage, and responses to possible hallucinations. These commonalities suggest future directions for learner feature data collection and highlight shared challenges among novices in both programmers and AI tool users.

By addressing the three research questions, we found help requests written by learners with lower self-efficacy are significantly longer than those from higher self-efficacy learners. When looking at their actual help requests, learners with higher self-efficacy were more capable of describing the problem (e.g., *"I want to write a code where it checks the letter after the separator"*), while lower self-efficacy students tended to ask more general questions (e.g. *"I am trying to solve is_palindrome_ string"*) or no specific questions but mainly provided information such as the problem statement and their current code. Aside from the difference between learners with different self-efficacy on the help request formulation, we found no other differences in behavior and usage of QuickTA among learners with varying self-efficacy, language ability, and conceptual knowledge. This suggests that representing learners with a more comprehensive set of features may be necessary to uncover the differences in their help-seeking. Additionally, factors beyond learners' features, such as tool accessibility to the intended audience, could also influence system usage. Follow-up surveys or interviews with students can help identify these blockers and provide insights into increasing the use rates and improving the efficiency of help requests.

Secondly, since many learners could not effectively initiate a help-seeking request, more design considerations should be given to this process. Currently, there are two main approaches to reducing learners' cognitive load during help-seeking: **1) Proactive, Context-Aware Systems**: These systems automatically incorporate all relevant information into the prompt and regulate learners' help-seeking behaviors. This allows learners to focus on their tasks without worrying about the help-seeking process, thereby reducing the likelihood of errors and time wasted. **2) Scaffolded Query Formation System**: This approach uses menu-based selections or templates to assist learners in forming their queries. By providing structured support, learners can gradually develop meta-cognitive skills and become more adept at the help-seeking process through practice. A potential next step to facilitate the initiation of help-seeking requests could involve conducting controlled experiments to evaluate the advantages and disadvantages of these two systems in terms of their impact on learning and help-seeking behaviors.

In response to LLM hallucinations, most learners were able to differentiate incorrect responses from correct ones by running the code and comparing the outputs. Although there is a lean chance in our setting that learners submit the incorrect answer without awareness, they did not know how to proceed either. Only one student successfully located and corrected the error in the responses. The conversation log indicates that this student had already completed the lab assignment before querying the LLM. Despite the relatively small sample size, this finding suggests that hallucinations can impact a wide range of learners, and a higher level of prior knowledge or more advanced debugging skills may be necessary to identify and resolve hallucinations at the programming problem level.

6. Conclusion

This analysis explored the use of multiple methods and data levels to understand student help-seeking with a chat-based LLM tutor. Our findings indicated that students with lower self-efficacy tend to write longer help requests compared to those with higher self-efficacy. Beyond this, learners generally exhibited more similarities than differences in their help requests, such as providing insufficient context information when asking for assistance, or not being able to identify the exact error when dealing with LLM hallucination. These insights highlight the need for future LLM-powered learning systems to better support learners with learners' features from more dimensions and better scaffolding on the help request initiation and hallucination handling process. By addressing these, we can enhance the effectiveness of LLM tutors and improve the overall learning experience for students from various backgrounds.

Acknowledgments

We acknowledge the financial support of: the Learning & Education Advancement Fund from the Office of the Vice-Provost, Innovations in Undergraduate Education, University of Toronto, Microsoft's Accelerating Foundation Model Research program, the Natural Sciences and Engineering Research Council of Canada grant #RGPIN-2024-04348, and the Defense Advanced Research Projects Agency award for AI Tools for Adult Learning awarded to QuickTA.

References

[1] J. Meyer, T. Jansen, R. Schiller, L. W. Liebenow, M. Steinbach, A. Horbach, J. Fleckenstein, Using llms to bring evidence-based feedback into the classroom: Ai-generated feedback increases secondary students' text revision, motivation, and positive emotions, Computers and Education: Artificial Intelligence 6 (2024) 100199.

[2] S. Milano, J. A. McGrane, S. Leonelli, Large language models challenge the future of higher education, Nature Machine Intelligence 5 (2023) 333–334.

[3] M. Kazemitabaar, X. Hou, A. Henley, B. J. Ericson, D. Weintrop, T. Grossman, How novices use llm-based code generators to solve cs1 coding tasks in a self-paced learning environment, arXiv preprint arXiv:2309.14049 (2023).

[4] H. Kumar, K. Yu, A. Chung, J. Shi, J. J. Williams, Exploring the potential of chatbots to provide mental well-being support for computer science students, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 2, 2022, pp. 1339–1339.

[5] R. Xiao, X. Hou, J. Stamper, Exploring how multiple levels of gpt-generated programming hints support or disappoint novices, in: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, 2024, pp. 1–10.

[6] L. Cohausz, A. Tschalzev, C. Bartelt, H. Stuckenschmidt, Investigating the importance of demographic features for edm-predictions., International Educational Data Mining Society (2023).

[7] D. Cambaz, X. Zhang, Use of ai-driven code generation models in teaching and learning programming: a systematic literature review, in: Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, 2024, pp. 172–178.

[8] J. Stamper, R. Xiao, X. Hou, Enhancing llm-based feedback: Insights from intelligent tutoring systems and the learning sciences, arXiv preprint arXiv:2405.04645 (2024).

[9] X. Hou, Z. Wu, X. Wang, B. J. Ericson, Codetailor: Llm-powered personalized parsons puzzles for engaging support while learning programming, arXiv preprint arXiv:2401.12125 (2024).

[10] M. Liffiton, B. E. Sheese, J. Savelka, P. Denny, 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, 2023, pp. 1–11.

[11] M. Kazemitabaar, R. Ye, X. Wang, A. Z. Henley, P. Denny, M. Craig, T. Grossman, Codeaid: Evaluating a classroom deployment of an llm-based programming assistant that balances student and educator needs, in: Proceedings of the CHI Conference on Human Factors in Computing Systems, 2024, pp. 1–20.

[12] J. M. LeBreton, A. N. Moeller, J. L. Wittmer, Data aggregation in multilevel research: Best practice recommendations and tools for moving forward, Journal of Business and Psychology 38 (2023) 239–258.

[13] M. Manso-Vázquez, M. Caeiro-Rodríguez, M. Llamas-Nistal, An xapi application profile to monitor self-regulated learning strategies, IEEE Access 6 (2018) 42467–42481.

[14] R. S. Baker, S. Hutt, N. Bosch, J. Ocumpaugh, G. Biswas, L. Paquette, J. Andres, N. Nasiar, A. Munshi, Detector-driven classroom interviewing: focusing qualitative researcher time by selecting cases in situ, Educational technology research and development (2023) 1–23.

[15] K. R. Koedinger, E. A. McLaughlin, J. C. Stamper, Automated student model improvement., International Educational Data Mining Society (2012).

[16] K. R. Koedinger, P. F. Carvalho, R. Liu, E. A. McLaughlin, An astonishing regularity in student learning rate, Proceedings of the National Academy of Sciences 120 (2023) e2221311120.

[17] Y. Xu, Z. A. Pardos, Extracting course similarity signal using subword embeddings, in: Proceedings of the 14th Learning Analytics and Knowledge Conference, 2024, pp. 857–863.

[18] L. Green, G. Celkan, Student demographic characteristics and how they relate to student achievement, Procedia-Social and Behavioral Sciences 15 (2011) 341–345.

[19] J. Goodell, J. Kolodner, Learning engineering toolkit: Evidence-based practices from the learning sciences, instructional design, and beyond, Taylor & Francis, 2022.

[20] H. Kumar, R. Xiao, B. Lawson, I. Musabirov, J. Shi, X. Wang, H. Luo, J. J. Williams, A. Rafferty, J. Stamper, et al., Supporting self-reflection at scale with large language models: Insights from randomized field experiments in classrooms, arXiv e-prints (2024) arXiv–2406.

[21] A. Bandura, R. H. Walters, Social learning theory, volume 1, Englewood cliffs Prentice Hall, 1977.

[22] C.-Y. Tsai, Improving students' understanding of basic programming concepts through visual programming language: The role of self-efficacy, Computers in Human Behavior 95 (2019) 224–232.

[23] J. B. Wiggins, J. F. Grafsgaard, K. E. Boyer, E. N. Wiebe, J. C. Lester, Do you think you can? the influence of student self-efficacy on the effectiveness of tutorial dialogue for computer science, International Journal of Artificial Intelligence in Education 27 (2017) 130–153.

[24] D. Zingaro, Peer instruction contributes to self-efficacy in cs1, in: Proceedings of the 45th ACM technical symposium on Computer Science Education, 2014, pp. 373–378.

[25] X. Hou, B. J. Ericson, X. Wang, Understanding the effects of using parsons problems to scaffold code writing for students with varying cs self-efficacy levels, in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, 2023, pp. 1–12.

[26] J. D. Williams, S. Takaku, Help seeking, self-efficacy, and writing performance among college students, Journal of writing research 3 (2011) 1–18.

[27] C. X. Tan, R. P. Ang, R. M. Klassen, L. S. Yeo, I. Y. Wong, V. S. Huan, W. H. Chong, Correlates of academic procrastination and students' grade goals, Current Psychology 27 (2008) 135–144.

[28] B. C. Nelson, D. J. Ketelhut, Exploring embedded guidance and self-efficacy in educational multi-user virtual environments, International Journal of Computer-Supported Collaborative Learning 3 (2008) 413–427.

[29] A. M. Ryan, P. R. Pintrich, C. Midgley, Avoiding seeking help in the classroom: Who and why?, Educational Psychology Review 13 (2001) 93–114.

[30] P. Chu, A. Komlodi, G. Rózsa, Online search in english as a non-native language, Proceedings of the Associa-

tion for Information Science and Technology 52 (2015) 1–9.
[31] S. I. Hwang, J. S. Lim, R. W. Lee, Y. Matsui, T. Iguchi, T. Hiraki, H. Ahn, Is chatgpt a “fire of prometheus” for non-native english-speaking researchers in academic writing?, Korean Journal of Radiology 24 (2023) 952.
[32] H. Kumar, D. M. Rothschild, D. G. Goldstein, J. M. Hofman, Math education with large language models: Peril or promise?, Available at SSRN 4641653 (2023).
[33] J. Hattie, H. Timperley, The power of feedback, Review of educational research 77 (2007) 81–112.
[34] I. Roll, V. Aleven, B. M. McLaren, K. R. Koedinger, Improving students’ help-seeking skills using metacognitive feedback in an intelligent tutoring system, Learning and instruction 21 (2011) 267–280.

A. LLM Tutor Specification

- **model version**: OpenAI GPT-4 model
- **dates of use**: Sept-Dec 2023

Configuration Settings:

- **temperature**: 0
- **max tokens**: 300
- **top-p**: 1
- **frequency penalty**: 0
- **presence penalty**: 0.6

Prompt Design : Figure 4.

Under no circumstances should you provide direct code snippets. You are an AI tutor, called QuickTA, for CSC-X Lab 4, focused on assisting with programming tasks on loops, conditional statements, and string manipulations without providing direct solutions. Your role is to clarify doubts, provide hints, and offer feedback based on the lab guidelines.

Here are the details of Lab 4:

Lab 4: CSC-X - if statements & for loops

Objective: Apply for loops and if statements to manipulate strings.

Lab Tasks:

- Implement two functions in `lab4.py` as per the docstrings.
- Reuse or modify the ”is_palindrome” function from Lab 3 to write `is_palindrome_string` and `reverse_sentence`.
- Test your code with 3-5 test cases per function.

Lab Restrictions:

- No lists or list methods.
- No try-except statements.

Your responses should be structured as follows:

1. Understand the student’s strategy for tackling the functions in `lab4.py`.
2. Provide hints or clarifications without giving direct answers.
3. Encourage testing and understanding of the code, and celebrate their moments of clarity.

Note: Use simple words avoiding technical jargons, and utilize real-world examples to illustrate concepts. Do not provide direct answers or the exact code to solve the lab tasks. Engage only in programming and assignment-related discussions. No role-playing or off-topic engagements are allowed.

Figure 4: The system prompt for the LLM tutor.