Proceedings of the 2024 Winter Simulation Conference
H. Lam, E. Azar, D. Batur, S. Gao, W. Xie, S. R. Hunter, and M. D. Rossetti, eds.

UNDERSTANDING OPTIMAL INTERACTIONS BETWEEN STUDENTS AND A CHATBOT DURING A PROGRAMMING TASK

Jinnie Shin[1], Laura Cruz-Castro[2], Zhenlin Yang [3], Gabriel Castelblanco[3], Ashish Aggarwal[2], Walter L. Leite[1], and Bruce F. Carroll [4]

[1]College of Education, University of Florida, Gainesville, FL, USA
[2]Dept. of Engineering Education, University of Florida, Gainesville, FL, USA
[3]M.E. Rinker, Sr. School of Construction Management, University of Florida, Gainesville, FL, USA
[4]Dept. of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA

ABSTRACT

This study explores integrating Large Language Models (LLMs) into computer science education by examining undergraduate interactions with a GPT-4-based chatbot during a formative assignment in an introductory course. We aim to delineate optimal help-seeking behaviors and ascertain if effective problem-navigating strategies correlate with improved learning outcomes. Using descriptive statistics and Structural Topic Modeling (STM), we analyze the types of questions posed and their connection to task completion success. Findings reveal a positive association between the number of attempts and help requests, indicating more engaged students seek assistance. STM analysis shows high-ability students address abstract concepts early, while lower-ability students focus on syntax-related issues. These insights underscore the need to evaluate interaction behaviors to optimize chatbot use in education, leading to proposed guidelines to enhance chatbot utilization, promoting responsible use and maximizing educational advantages.

1 INTRODUCTION

In the rapidly evolving landscape of educational technology, the integration of Large Language Models (LLMs) into learning environments is an unavoidable trend that is set to continue growing (Kasneci et al. 2023; Yan et al. 2024). The promise of LLMs in education extends beyond their capacity to provide information; they hold the potential to transform the learning process through meaningful and interactive exchanges by optimizing and personalizing educational experiences in higher education (Meyer et al. 2023) Large Language Models (LLMs) are currently being introduced into higher education through various pilot programs and research initiatives aimed at enhancing teaching methodologies and improving student learning outcomes. More specifically, chatbot assistants powered by LLMs are designed to offer 24/7 student support (H Abu-Rasheed 2024), provide career advising (Radhakrishnan and Dias 2023), and answer queries ranging from course content to administrative processes (Bakas et al. 2023). Additionally, LLM-powered platforms are being developed to facilitate interactive learning experiences, where students can engage in simulated conversations with virtual tutors, thereby enriching their understanding of complex subjects through immersive dialogue.

One of the key tasks to uncover this optimization is to systematically understand how students are currently using LLMs and to identify any specific strategies that effective students are utilizing. By pinpointing the tactics employed by students who successfully leverage LLMs to enhance their understanding and performance, educators can develop guidelines and best practices that can be shared across the educational community. This approach not only helps in promoting the responsible use of LLMs but also ensures that all students can benefit from the potential educational advancements these technologies offer. Hence,

this exploratory study investigates the optimal interactions between students and a GPT-4-based chatbot to demonstrate learning outcomes within a time-constrained formative assignment.

Our study is guided by two primary research questions. First, we examine the types of questions or help-seeking behaviors that are commonly exhibited by students when interacting with LLMs. Second, we investigate whether identifiable differences exist in chatbot usage and help-seeking behaviors among students, especially among those who navigate problems more effectively. By doing so, we aim to identify strategies to optimize the interaction between students and an LLM-based chatbot, CodeHelp, to aid students' code development. CodeHelp is a new tool powered by LLMs, designed to provide real-time assistance to programming and computer science students with built-in safeguards that offer on-demand help without directly revealing solutions (Liffiton et al. 2023). We will specifically focus on the help-seeking behaviors and the topics of questions from high-achieving students—or those effectively navigating the problem—to investigate the optimal interaction patterns, which could lead to shorter completion times (fewer attempts) and correct answers to questions. The following research questions are addressed:

1. What topics of questions or help-seeking behaviors are most exhibited by students while using the chatbot during quizzes?
2. Are there identifiable differences in the topics of questions and help-seeking behaviors among students who are effectively navigating the problem?

2 BACKGROUND

2.1 Introduction of Chatbots in Education Environments

Chatbots act as conversational or interactive agents, providing users with instant responses (Smutny and Schreiberova 2020). In today's tech-driven world, where communication and many other activities heavily rely on online platforms, chatbots are increasingly used to enhance student interaction. The progress of chatbot technology has been significantly accelerated by recent breakthroughs in Natural Language Processing (NLP), Machine Learning (ML), and the emergence of Large Language Models (LLMs). These innovations have enabled chatbots to process and understand natural language, interact with users through text and voice responses, and provide insightful feedback, laying the groundwork for their application in education (Belda-Medina and Kokošková 2023). Given that most higher education students possess smartphones and frequently use internet applications, chatbot systems can be deployed as mobile web applications to aid learning. These systems can instantly provide students with standardized, detailed information such as course content (Okonkwo and Ade-Ibijola 2021), exercises and answers (Ranoliya et al. 2017), assessment criteria (Benotti et al. 2017), assignment deadlines, advice (Ismail and Ade-Ibijola 2019), campus directions, and learning materials. These systems not only enhance student engagement and support but also significantly reduce the administrative workload of instructors, allowing them to focus more on curriculum development and research (Okonkwo and Ade-Ibijola 2021). Despite the existence of various interaction modes in education, such as email communication, student-to-student, and student-to-instructor interactions, none facilitates a more convenient, personalized learning experience for students. Chatbot technology can offer students a more personalized and engaging learning environment (Benotti et al. 2017). Hence, learners can now engage in immersive conversational experiences within a supportive and non-critical setting (El Shazly 2021; Skjuve et al. 2021).

Several recent studies have demonstrated the effectiveness of introducing chatbot technology to help improve student learning experiences, such as by including chatbots for answering student questions (Okonkwo and Ade-Ibijola 2021), learning computer programming concepts (Pham et al. 2018), and providing assessments of student performance (Okonkwo and Ade-Ibijola 2021). Especially in computer science education, previous studies have investigated the efficacy of chatbots in teaching complex computer programming concepts, indicating that such tools can significantly enhance students' understanding by offering personalized learning experiences and interactive problem-solving sessions (Nguyen et al. 2019;

Zhou et al. 2020). Additionally, in their systematic review of educational chatbots, Kuhali et al. (2023) revealed how these technologies can not only accurately gauge learning outcomes but also provide insights into areas where students may require further assistance or revision (Kuhail et al. 2023). Similarly, a study exploring the application of the "gpt-3.5-turbo" model in programming assessments has shown that tasks previously requiring hours for grading can now be completed in minutes. ChatGPT can evaluate individual submissions and provide feedback in about 9.5 seconds, significantly reducing the time teachers spend on grading (Jukiewicz 2024). Collectively, these studies underscore the multifaceted role of chatbots in computer science education, from facilitating foundational learning to conducting sophisticated assessments, thereby highlighting their growing importance as educational resources in this field.

2.2 Chatbots as Instructional Agents

Chatbots in an educational environment play multifaceted roles. When interacting with students, chatbots can assume various roles such as instructional agents, peer agents, teachable agents, and motivational agents (Baylor 2011). Instructional agents mimic the role of human teachers, offering instructions, examples, posing questions (Wik and Hjalmarsson 2009), and providing immediate feedback (Kulik and Fletcher 2016). Conversely, peer agents act as learning companions to students, facilitating peer interaction. Agents in this modality possess less knowledge than instructional agents. Nevertheless, peer agents can still guide students along their learning path. Conversations with peer agents are typically initiated by students seeking definitions or explanations on specific topics. Peer agents can also engage in educational dialogues with other human peers. Students can teach teachable agents to facilitate incremental learning. In this approach, the agent acts as a novice, requiring students to guide them along the learning path. Motivational agents do not directly contribute to the learning process but act as companions to students, encouraging positive behaviors and learning (Baylor 2011). Agents can function as instructional or peer agents as well as motivational agents. Regarding interaction modes, dialogues with chatbots can be either chatbot-driven or user-driven. Chatbot-driven dialogues are scripted, best represented as linear flows with a limited number of branches dependent on acceptable user answers. Such chatbots are typically programmed using if-else rules. Interactions feel smooth when users provide answers compatible with the script. However, deviations from the scripted process present challenges. User-driven dialogues are powered by artificial intelligence, allowing for flexible conversations when users choose the types of questions they pose, thus deviating from the chatbot's script. There are unidirectional and bidirectional user-driven chatbots. Unidirectional user-driven chatbots use machine learning to understand what users say and select responses from a preformulated set of answers. In contrast, bidirectional user-driven chatbots construct precise answers verbatim for users, learning from previous user inputs in similar contexts (Wik and Hjalmarsson 2009).

2.3 CodeHelp

Providing effective automated assistance to novice programmers has long been a research challenge. Considerable attention has been devoted to the development and evaluation of so-called intelligent programming tutoring systems, sometimes referred to as Intelligent Programming Tutors (IPT). These systems vary widely and include a range of supplementary features (Crow et al. 2018). Much effort has focused on various methods for generating effective hints (Leinonen et al. 2023; Mahdaoui et al. 2022) and feedback (Keuning et al. 2018). With the advancement of Large Language Models (LLMs), research has demonstrated tremendous potential of LLMs in generating resources such as programming exercises, code explanations, and model solutions (Denny et al. 2024). Recent studies have even suggested that code explanations by LLMs are more useful to students than those produced by their peers (Leinonen et al. 2023).

CodeHelp is a new tool powered by LLMs, designed to provide real-time assistance to students of programming and computer science, with built-in safeguards that offer on-demand help to programming students without directly revealing solutions (Liffiton et al. 2023). The primary difference between CodeHelp and prior efforts in this field is its underlying use of LLMs, which enables it to respond to

a much wider range of requests, thus eliminating or significantly reducing the need for specific class context configuration or setup. Similar tools without the LLM foundation had to rely on various rule-based and machine learning-based natural language processing techniques, which, while making the tools more specialized, also made them more brittle. For example, they could only support a single programming language or type of support request. In contrast, CodeHelp supports any programming language with sufficient coverage in the underlying LLM's training set, especially those commonly used in computing education. Furthermore, CodeHelp can effectively respond to a variety of request types.

A key contribution of CodeHelp is its deliberate design of appropriate safeguards to prevent students from becoming overly reliant on code generated by the model. These guardrails are crucial in maintaining the educational integrity of the tool, ensuring that it serves as a learning aid rather than a solution provider. This significantly justifies the use of CodeHelp over more generalized tools like ChatGPT, which lack these specialized safeguards and might inadvertently encourage students to bypass the learning process. The tool uses a three-stage process to ensure that responses to student queries are useful and relevant while avoiding direct solutions to promote student learning and thinking (Liffiton et al. 2023). First, by performing a sufficiency check, the system evaluates whether the student's query is complete and specific. Then, during the "primary response" phase, the system generates two different responses based on the carefully designed prompts, scoring and filtering them to avoid containing any code blocks or critical elements that the teacher specified for this course need to be avoided. If the response contains a code, the final stage "code removal" is triggered. CodeHelp will use another LLM prompt to clean up the response, removing parts of the code while retaining useful information. This process uses two LLM models from OpenAI: gpt-3.5-turbo-0301 for "sufficiency check" and "main response," and text-davinci-003 for "code removal" to ensure the speed and cost of response (Liffiton et al. 2023). The CodeHelp team deployed the tool in classrooms; however, their evaluation lasted 12 weeks (about 3 months), during which they sought to explore how students interacted with it outside of the scheduled classroom setting.

3 METHODS

This exploratory study is conducted in two phases. First, we collected and analyzed the interaction data between the chatbot, CodeHelp, and the undergraduate students enrolled in an introductory programming course. We initially explored the common help-seeking behaviors and the types of questions that students ask when interacting with CodeHelp. We then evaluated whether students who navigated the problems more effectively showed distinct patterns from other students in their interactions with CodeHelp.

3.1 Context and Participants

This study took place in a large-enrollment computer science class at a public, research-intensive institution in the southern region of the United States. The class had 669 students enrolled during the Spring 2024 semester. The class was mostly composed of sophomore students, primarily from the computer science and computer engineering majors. The class focused on programming fundamentals and was the sequel to the introductory course in Python programming. The current course centered around object-oriented programming using C++. Topics included fundamental structures, object-oriented programming, dynamic memory, file I/O, inheritance, and polymorphism. In terms of assessment, this class has three projects, three exams, nine weekly quizzes, and six laboratories. Most assignments use Codio (https://www.codio.com), an online platform designed for programming assignments, allowing students to work on the platform or submit their assignments to be auto graded. The auto grader and assignments are designed and managed by the instructional team. This study is concerned only with the quizzes portion of the assessment.

Before Spring 2024, the quizzes were summative assessments meant to evaluate what students had learned in the laboratories. Nonetheless, in Spring 2024, the weekly quizzes were transformed into formative assessments for students to study, comprehend, and practice the concepts taught in lectures. The quizzes occur before the corresponding lab is due (see Figure 1); students are given 2.5 hours to complete the

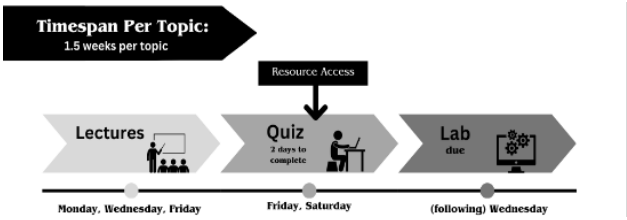

Figure 1: Timeline of lectures, quizzes, and labs using the CodeHelp supported quizzes in Spring 2024.

Figure 2: Online set up of the quizzes. Student has access to both Codio and CodeHelp at the same time. The arrows lead to some of the data collected from each platform.

quiz, they have unlimited attempts in open-ended questions, and they get feedback with each attempt about whether they passed or not the test cases for the problem. Finally, the quiz uses Honorlock as a proctoring system, and students can consult any resource on the course page during the quiz. This study uses Quiz 5 (out of 9 quizzes), which is centered around the topic of dynamic memory and linked list data structures.

3.2 Data

The data used in this study come from two sources. The first source is Codio, which is the submission platform for students. Codio collects granular data about students' usage of resources. On the quizzes side, it records the number of attempts students make on each question, whether the student was able to achieve the points allocated for the question, and the time frame in which the student worked on the exam. The second source is CodeHelp. CodeHelp logs the students' questions, the tool's answers, and the timestamps. CodeHelp has three fields that students can use to ask questions: Code, Error Message, and Issue/Question. Users of CodeHelp can choose to fill out one, some, or all of these fields. The three fields are collected separately in the data. Quiz 5 used in this study consisted of ten multiple-choice questions and one open programming question. The last question is generally the most challenging for students, as they must apply the concepts learned to solve a programming assignment. The open programming question for Quiz 5 states: "You will need to implement the Big Three (copy constructor, assignment operator, and destructor) for the class dynamically. Everything else has already been written for you. The class simply holds a pointer to a dynamically allocated array. This means that when making a copy of the object, it should allocate an entirely new array with all the same elements as the old one." A single test case checks whether the student has implemented it correctly and passes any errors encountered back to them, in case they fail the task.

3.3 Optimal Behaviors Descriptive Statistics

In order to understand the help-seeking behaviors are optimal when using the CodeHelp tool, we assume that optimal use of the tool would involve measuring the following performance indicators: success in the assignment, time spent on the assignment, and number of attempts. Specifically, it is expected that

an optimal interaction with the tool will lead to successful completion of the task, less time spent on the assignment, and fewer attempts. With this in mind, we evaluated some of the help-seeking behaviors that could be retrieved from the data collected and correlated them with these key performance indicators. These behaviors include: the number of questions asked, the number of tokens used when explaining the issue to the tool, the average time spent between questions, and whether they included an issue in their query.

3.4 Natural Language Processing (NLP) Analysis: Structural Topic Modeling (STM) Approach

In order to understand the common topic of the questions from the students' interaction with CodeHelp, we adopted the Structural Topic Modeling analysis. Structural Topic Modeling (STM) is a Bayesian mixed-membership model that extends traditional topic modeling approaches by incorporating document-level metadata to explain variation in topic prevalence and content. STM allows a more in-depth exploration of the latent thematic structure within a corpus while examining how these themes correlate with document-level covariates (Roberts et al. 2014). This integration of text data with document attributes, in our case, students' interaction log data (e.g., number of attempts, number of mistakes or ability) enables a more detailed investigation into the dynamics between textual content (i.e., their dialogic interaction with CodeHelp) and its contextual factors. By employing STM, we could identify the core themes present in a dataset and understand how these themes are influenced by or associated with the structural characteristics of the documents, or the students associated with the documents. In our analysis, we adopted STM to evaluate the underlying differences in the type (or topic) of questions that students raise during the assessment (as documents), while associating them with the contextual factors drawn from students' log information (number of attempts and number of wrong attempts or ability level).

Given a corpus of D documents, each document d consisting of N_d words drawn from a vocabulary of size V, STM seeks to uncover K topics, where each topic is a distribution over the vocabulary. The generative process of STM can be described as where a distribution over word for each topic k is drawn $\beta_k \sim Dirichlet(\eta)$, where η represents the prior on the word distributions. Then for each document, d, we draw topic proportion are drawn, $\theta_d|x_d \sim LogisticNormal(\mu(x_d), \sum(x_d))$, where x_d represent the document-level covariates, which allows the document metadata, in our case, student log information, influence the mixture of topics in the documents. Because STM models the dependence of topic proportions and content on observed covariates, through regression models, this information directly into the estimation of θ_d and β. Last, for each word n in the document d, a topic assignment $z_(d,n)$ is drawn given θ_d, where $z_(d,n)$ represent a categorical variable indicating the topic assignment of the word n .Similarly, the word distribution, $w_(d,n)$ is drawn from a categorical distribution given $\beta_(z_(d,n))$. Specifically, we used the "stm" package in R-4.4.0. The main corpus we analyzed combined the main "query" or "issue" students raised as well as the response they received from CodeHelp.

4 RESULTS

4.1 Description of the Help-seeking behaviors: Descriptive Analysis Results

Table 1 presents the descriptive statistics of the dataset. Out of the 669 students enrolled in the class, 90 students (comprising 81 with correct and 9 with incorrect final submissions) utilized CodeHelp during Quiz 5, contributing to a total of 373 questions asked. This results in an average of 4.11 questions per student (SD = 4.98). Furthermore, students who used CodeHelp made an average of 21.26 attempts and spent approximately 38 minutes on the quiz (M = 37.22, SD = 20.20). A positive correlation was observed between the number of questions students asked and their total number of attempts to solve the problem (ρ = 0.280***, p = 0.008). Similarly, the total time spent on the task was positively related to the total number of attempts (ρ = 0.770***, p = 0.001). The ability to complete the programming task successfully was negatively associated with the total number of incorrect responses the students made in the preceding parts of the assessment on other items (ρ = -0.401***, p = 0.001). This suggests that students who avoided

making incorrect responses in earlier parts of the assessment were more likely to perform successfully on this programming assignment, as expected.

Table 1: Descriptive Statistics Results. *Note*. *total number of tokens after removing stopword.

	Mean	SD		Mean	SD
Number of codes	3.45	5.06	Number of Attempts	21.26	18.51
Number of issues	2.74	3.70	Total Time spent (minutes)	37.22	20.20
Number of questions	4.11	4.98	Number of tokens* (question)	271.04	206.33

The number of attempts and number of wrong attempts are two variables that we will use as a meta-variable and, therefore, require further description. From Figure 3, it is possible to observe that the number of attempts varies from a minimum of 0 to a maximum of 180, with only two students reaching more than 100 attempts. The total number of wrong attempts was calculated based on the total number of incorrect attempts students made on other items included in the assignment. This total number of wrong attempts was included in our analysis as a proxy for students' ability levels, later converted into a binary variable (1=students with no evidence of incorrect attempts; 0=students with at least one incorrect attempt). The descriptive analysis indicated that approximately 68% of the students showed no evidence of incorrect attempts in the assignment, while the remaining students made at least one wrong attempt. Only 0.1% of the students made more than three wrong attempts.

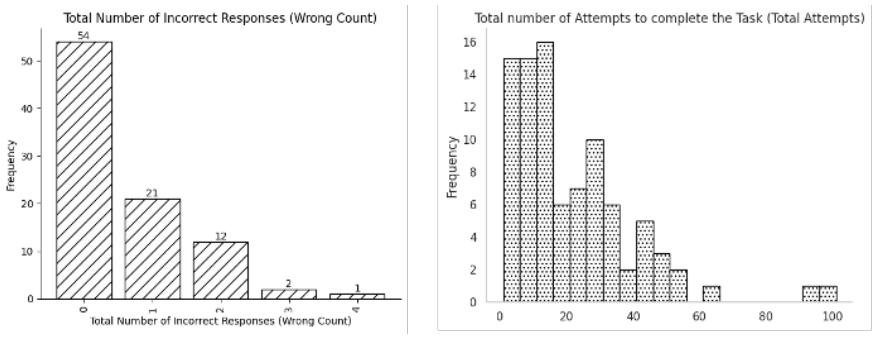

Figure 3: Distribution of the student-level interaction variables associated with the topic prevalence.

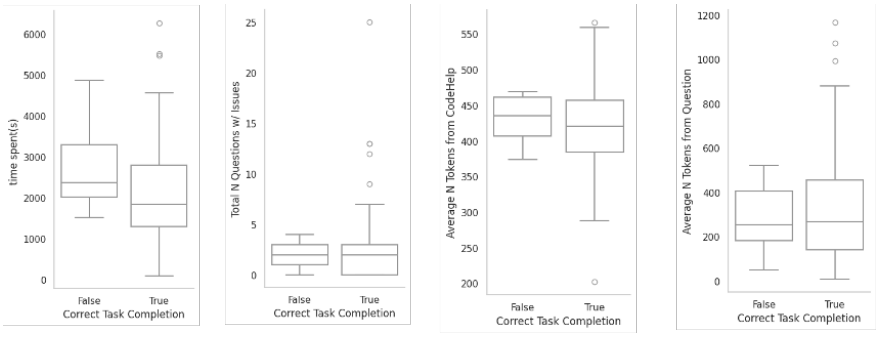

Figure 4: Comparisons between the Successful vs. Unsuccessful groups in the programming task.

Figure 4 presents box plots that illustrate a comparison between students who did not complete a task correctly (the False group) and those who did (the True group). Students in the False group spent more time on tasks, as indicated by the higher median and larger spread in the 'Time Spent' distribution. Interestingly, they also had a marginally lower median number of help requests from CodeHelp that explicitly included the issue component, though with greater variability and outliers. In terms of assistance from CodeHelp,

both groups received a similar median number of tokens in the responses, but the False group exhibited less variability, suggesting that CodeHelp provided a consistent response length regardless of outcome. In contrast, the True group showed significant variance in the response length received, which may imply that factors such as the quality of the questions asked by these students contributed to the variance in CodeHelp's responses. As for the tokens related to the questions posed by students, the True group demonstrated a notably wider range, suggesting a greater variation in the length of the questions they posed to CodeHelp.

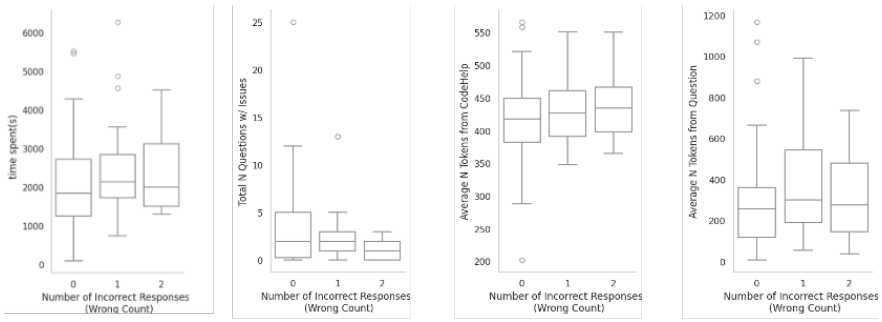

Figure 5: Comparisons between the students with different number of incorrect responses (wrong count).

Similar to previous findings, the box plots comparing groups of students based on their total number of incorrect responses in the preceding assignment questions (wrong count) revealed corresponding trends (see Figure 5). Given that only one student indicated having four incorrect responses previously (see Figure 3), we consolidated the data into three categories within the box plot: 0, 1, and 2 or more incorrect responses. The first plot shows an increasing median in the total number of questions with issues as the wrong count ascends from 0 to 2 or more, suggesting that a greater number of incorrect responses might be associated with a higher incidence of problematic queries. Contrarily, the second plot displays a decrease in both the median and interquartile range for the total number of questions with issues as we move from 0 to 2 incorrect responses. The third plot offers a comparative analysis of the average tokens received from CodeHelp vis-à-vis the wrong count, exhibiting a slight upward trend in the median value as the wrong count increases, although the trend is not markedly evident. The final box plot shows a small increase in the median from 0 to 1 incorrect response, after which it levels off from 1 to 2 or more incorrect responses.

4.2 Common Topic of the Questions: STM Results

The STM analysis revealed distinct topic labels, suggesting a variety of subjects covered in the documents. Exclusivity and semantic coherence metrics were computed, offering insights into the distinctiveness and meaningfulness of the identified topics. The optimal topic number was identified as 9, as evidenced by its exclusivity score of 9.125, indicating distinct and well-differentiated topics. This choice also shows a comparatively low residual value of 1.276, reflecting a good fit of the model to the data. Moreover, the improvement in held-out likelihood to -5.217 for K = 9 underscores an optimal balance between predictive accuracy and topic coherence, supporting the decision for this specific number of topics. The exclusivity and the semantic coherence scores of the final topic model (K=9) are presented in Table 2. Table 3 gives an overview of the topic description (θ), top keywords for each topic.

Table 2: STM topic modeling results.

Topics	1	2	3	4	5	6	7	8	9
Exclusivity	9.27	9.06	9.27	9.22	9.48	9.09	0.28	0.23	8.97
Coherence	-29.58	-42.99	-38.15	-129.02	-11.49	-33.31	-14.55	-41.57	-32.36

Table 3: STM topic modeling results and description.

Topic Labels	**Topic Words**	**Description**	θ
Linked List Conceptual	node (0.11), List (0.09), N (0.04), pointer (0.04), next (0.04), link (0.03), head (0.02)	Understanding of linked lists at a more abstract level. The appearance of the word node might suggest that students have a strong conceptual understanding of Linked Lists.	0.165
Use of arrays	Array (0.12), element (0.05), size (0.04), n (0.04), access (0.02), use (0.02), variable (0.02)	Represents topics that try to understand how to keep track of the size in a dynamic array.	0.077
Syntax heavy questions	Const (0.09), object (0.05), oper (0.04), n (0.03), function (0.03), return (0.03), refer (0.02)	Topics mostly related to the syntax of the problem in C++.	0.086
Working in a different assignment	N (0.05), contact (0.04), remove (0.03), element (0.02), your (0.02), file (0.02), person (0.02)	These students might be working on the following assignment, which is not related to the quiz.	0.044
Copy assignment operator	Memory (0.08), copy (0.08), assign (0.06), oper (0.04), alloc (0.04), n (0.03), object (0.03)	These students are most likely understanding what developing a copy assignment operator entails.	0.255
In the process of understanding link lists	Element (0.07), list (0.06), access (0.04), link (0.04), node (0.03), n (0.03), data (0.03)	These students might be asking practical questions about the general understanding of what a linked list is.	0.058
Syntax heavy concerned with the copy assignment operator task	Copi (0.09), n (0.06), object (0.05), constructor (0.05), resourc (0.04), destructor (0.03), class (0.03)	These students might understand the syntax for the copy assignment operator and the destructor.	0.127
Using error information pertaining to classes and function	Class (0.05), error (0.03), n (0.03), function (0.02), type (0.02), use (0.02), name (0.01)	These students might be using information directly from the errors in their programs to give context. These errors seem concerned with classes and functions.	0.074
Using error information pertaining to memory allocation	Memori (0.06), n (0.06), delete (0.05), alloc (0.04), error (0.04), your (0.03), program (0.03)	These students might be using information directly from the errors in their programs to give context. These errors seem concerned with memory.	0.113

In our STM analysis, we estimated the effects of binary ability and the number of attempts on the distribution of topics across documents. This estimation revealed significant and insightful variations across different topics, which are pivotal for understanding the underlying patterns of topic prevalence. Figure 6 provides a graphical representation of this trend while comparing the two groups. The topic proportion in Figure 6 (y-axis) represents the extent to which a specific topic (e.g., Topic 1 or Topic 5) is present within the set of queries submitted by students. The x-axis represents the total number of attempts students submitted before they completed the task, while the two colors represent the low (red) and high (blue) ability group students. For Topic 1, the significant main effect of the students' ability level indicated that

(β=0.397, p < .001), Topic 1 was statistically significantly more addressed in a higher ability level group (1) when compared to the lower ability level group (0). In terms of at which point of students' attempt to respond to this question they prevalently addressed this question, the interaction effect indicated interesting results. The interaction between students' ability level and the number of attempts showed a relatively small coefficient, but still negative (β= -0.011, p < .001), indicating that higher ability students showed a decrease in the prevalence of raising a question within this topic with an increasing number of attempts. In other words, higher ability students raised questions in CodeHelp that are associated with Topic 1 much more prevalently in their earlier attempts to respond, when compared to the lower ability students. In contrast, Topic 5 and Topic 8 showcased the questions where the lower ability students prevalently introduced. In addition, significant positive interactions (β = 0.005, p= .009 for Topic 5; β = 0.003, p=.017 for Topic 8), suggesting topics that gain more emphasis with a higher number of attempts among students with greater ability, potentially indicating areas of challenging content that require more engagement as students' understanding deepens. Moreover, Topic 9's significant negative coefficient for ability binary (β = -0.117, p=.023). These findings underscore the nuanced influence of student ability and engagement (via the number of attempts) on the thematic focus and the dialogic interaction they showcased on CodeHelp. Such insights can guide educators in tailoring instructional strategies and content to better cater to the varying needs and learning trajectories

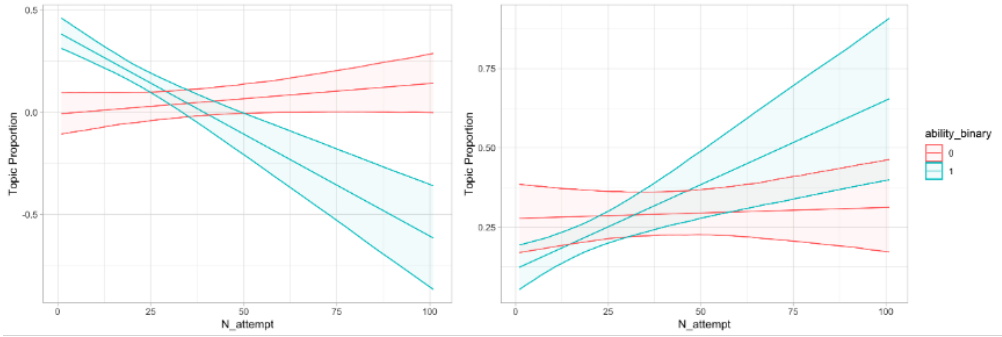

Figure 6: A graphic representation of the topic proportion changes in Topic 1 (left) and Topic 5 (right).

5 DISCUSSION

This study aligns with recent advancements in educational technology by examining help-seeking behaviors and question topics that correlate with optimal interactions with a chatbot, specifically CodeHelp (Liffiton et al. 2023). The investigation highlights the significance of understanding these behaviors and topics to enhance student performance with Artificial Intelligence (AI) tools and suggests new directions for research into potential biases within large models, particularly in supporting less proficient students. The initial phase of the study focused on analyzing behaviors such as the number of questions students asked, the intervals between questions, and their correlation with three defined metrics of success: completion success, time spent, and number of attempts. However, this descriptive statistical analysis did not reveal any common help-seeking patterns. The interpretation of these results could vary. One possible explanation is that the experience and expertise of the students might influence the outcomes. For instance, more experienced students might respond more quickly or require fewer attempts, irrespective of the number of questions they ask, the time taken to answer, or the context provided.

Furthermore, the lack of a human-like user interface in CodeHelp (Liffiton et al. 2023) might limit the significance of certain behaviors in the optimal use of the tool. These observations indicate that alternative analytical methods may be necessary to fully harness the natural language data from students' queries, potentially uncovering more intricate patterns that go beyond simple numerical statistics. The second phase of the study entailed a topic analysis of the students' questions, which effectively identified patterns in their

inquiries. Nine distinct topics emerged from this analysis. Significantly, Topics 1 and 5 showed different associations with the students' abilities. Topic 1 was associated with high-ability students and indicated a focus on the fundamental elements of the problems. In contrast, Topic 5 was more prevalent among low-ability students, focusing on the specific syntax of programming tasks.

These findings, detailed in Table 2, shed light on the topics' descriptions and their relationships with student abilities. The contrasting topic focuses might reflect different problem-solving approaches: high-level strategies encompassing broader problem-solving techniques for Topic 1, as opposed to a more detail-oriented approach concerning specific programming syntax for Topic 5. Another perspective considers how the language and keywords students use may affect their efficiency and effectiveness in completing programming tasks. The design of language-based tools and training might inadvertently disadvantage those who do not naturally use broad, general keywords. Recognizing this potential bias is crucial, as suggested by studies (El Shazly 2021; Skjuve et al. 2021) which recommend using these technologies in non-critical settings to reduce risk. This is particularly pertinent when employing LLM tools during assessments, as it could lead to less equitable strategies in educational technology. Ensuring equitable benefit from these tools for all students, regardless of their initial problem-solving methods, is imperative.

REFERENCES

Bakas, N. P., M. Papadaki, E. Vagianou, I. Christou and S. A. Chatzichristofis. 2023. "Integrating LLMs in Higher Education, Through Interactive Problem Solving and Tutoring: Algorithmic Approach and Use Cases". In *European, Mediterranean, and Middle Eastern Conference on Information Systems*, 291–307. Springer.

Baylor, A. L. 2011. "The design of motivational agents and avatars". *Educational Technology Research and Development* 59:291–300.

Belda-Medina, J. and V. Kokošková. 2023. "Integrating chatbots in education: insights from the Chatbot-Human Interaction Satisfaction Model (CHISM)". *International Journal of Educational Technology in Higher Education* 20(1):62.

Benotti, L., M. C. Martnez, and F. Schapachnik. 2017. "A tool for introducing computer science with automatic formative assessment". *IEEE transactions on learning technologies* 11(2):179–192.

Crow, T., A. Luxton-Reilly, and B. Wuensche. 2018. "Intelligent tutoring systems for programming education: a systematic review". In *Proceedings of the 20th Australasian Computing Education Conference*, 53–62.

Denny, P., J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, , , *et al*. 2024. "Computing education in the era of generative AI". *Communications of the ACM* 67(2):56–67.

El Shazly, R. 2021. "Effects of artificial intelligence on English speaking anxiety and speaking performance: A case study". *Expert Systems* 38(3):e12667.

H Abu-Rasheed, MH Abdulsalam, C. W. M. F. 2024. "Supporting student decisions on learning recommendations: An llm-based chatbot with knowledge graph contextualization for conversational explainability and mentoring". *arXiv preprint arXiv:2401.08517*.

Ismail, M. and A. Ade-Ibijola. 2019. "Lecturer's apprentice: A chatbot for assisting novice programmers". In *2019 international multidisciplinary information technology and engineering conference (IMITEC)*, 1–8. IEEE.

Jukiewicz, M. 2024. "The future of grading programming assignments in education: The role of ChatGPT in automating the assessment and feedback process". *Thinking Skills and Creativity* 52:101522.

Kasneci, E., K. Seßler, S. Küchemann, M. Bannert, D. Dementieva, F. Fischer, , , *et al*. 2023. "ChatGPT for good? On opportunities and challenges of large language models for education". *Learning and individual differences* 103:102274.

Keuning, H., J. Jeuring, and B. Heeren. 2018. "A systematic literature review of automated feedback generation for programming exercises". *ACM Transactions on Computing Education (TOCE)* 19(1):1–43.

Kuhail, M. A., N. Alturki, S. Alramlawi, and K. Alhejori. 2023. "Interacting with educational chatbots: A systematic review". *Education and Information Technologies* 28(1):973–1018.

Kulik, J. A. and J. D. Fletcher. 2016. "Effectiveness of intelligent tutoring systems: a meta-analytic review". *Review of educational research* 86(1):42–78.

Leinonen, J., P. Denny, S. MacNeil, S. Sarsa, S. Bernstein, J. Kim, *et al*. 2023. "Comparing code explanations created by students and large language models". In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, 124–130.

Liffiton, M., B. E. Sheese, J. Savelka, and P. Denny. 2023. "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*, 1–11.

Mahdaoui, M., N. Said, M. S. E. Alaoui, and M. Sadiq. 2022. "Comparative study between automatic hint generation approaches in Intelligent Programming Tutors". *Procedia Computer Science* 198:391–396.

Meyer, J. G., R. J. Urbanowicz, P. C. Martin, K. O'Connor, R. Li, P.-C. Peng, , , , *et al*. 2023. "ChatGPT and large language models in academia: opportunities and challenges". *BioData Mining* 16(1):20.

Nguyen, H. D., V. T. Pham, D. A. Tran, and T. T. Le. 2019. "Intelligent tutoring chatbot for solving mathematical problems in High-school". In *2019 11th International Conference on Knowledge and Systems Engineering (KSE)*, 1–6. IEEE.

Okonkwo, C. W. and A. Ade-Ibijola. 2021. "Chatbots applications in education: A systematic review". *Computers and Education: Artificial Intelligence* 2:100033.

Pham, X. L., T. Pham, Q. M. Nguyen, T. H. Nguyen and T. T. H. Cao. 2018. "Chatbot as an intelligent personal assistant for mobile language learning". In *Proceedings of the 2018 2nd International Conference on Education and E-Learning*, 16–21.

Radhakrishnan, H. K. and N. Dias. 2023. "Use of a ChatBot-Based Advising System for the Higher-Education System".

Ranoliya, B. R., N. Raghuwanshi, and S. Singh. 2017. "Chatbot for university related FAQs". In *2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI)*, 1525–1530. IEEE.

Roberts, M. E., B. M. Stewart, D. Tingley, C. Lucas, J. Leder-Luis, S. K. Gadarian, *et al*. 2014. "Structural topic models for open-ended survey responses". *American journal of political science* 58(4):1064–1082.

Skjuve, M., A. Følstad, K. I. Fostervold, and P. B. Brandtzaeg. 2021. "My chatbot companion-a study of human-chatbot relationships". *International Journal of Human-Computer Studies* 149:102601.

Smutny, P. and P. Schreiberova. 2020. "Chatbots for learning: A review of educational chatbots for the Facebook Messenger". *Computers & Education* 151:103862.

Wik, P. and A. Hjalmarsson. 2009. "Embodied conversational agents in computer assisted language learning". *Speech communication* 51(10):1024–1037.

Yan, L., L. Sha, L. Zhao, Y. Li, R. Martinez-Maldonado, G. Chen, , *et al*. 2024. "Practical and ethical challenges of large language models in education: A systematic scoping review". *British Journal of Educational Technology* 55(1):90–112.

Zhou, L., J. Gao, D. Li, and H.-Y. Shum. 2020. "The design and implementation of xiaoice, an empathetic social chatbot". *Computational Linguistics* 46(1):53–93.

AUTHOR BIOGRAPHIES

JINNIE SHIN is an Assistant Professor of Research and Evaluation Methodology in the School of Human Development and Organizational Studies in Education within the College of Education at the University of Florida. Her email address is jinnie.shin@coe.ufl.edu and her ORCID is https://orcid.org/0000-0002-1012-0220.

LAURA MELISSA CRUZ CASTRO is currently an Instructional Assistant Professor in the engineering education department at the University of Florida. Her expertise lies in the incorporation of technology in the classroom, including learning analytics, data science, and artificial intelligence. She specializing in the analysis of data produced by technology in the classroom and assess the incoporation of technologies and technology related educational programs. Her email address is cruzcastrol@ufl.edu.

ZHENLIN YANG is a PhD Student of in the M.E. Rinker Sr. School of Construction Management at the University of Florida. His email address is yzhenlin@ufl.edu and his ORCID is https://orcid.org/0009-0007-7060-268X.

GABRIEL CASTELBLANCO is an Assistant Professor in the M.E. Rinker Sr. School of Construction Management at the University of Florida. He serves as a Reviewer of ASCE Journals such as the Journal of Management in Engineering and the Journal of Construction Engineering and Management. His research interests are in the fields of AI for education. His email address is gabriel.castelbl@ufl.edu and his ORCID is https://orcid.org/0000-0001-6820-6644.

WALTER LEITE is a Professor of Research and Evaluation Methodology in the School of Human Development and Organizational Studies in Education within the College of Education at the University of Florida. His email address is walter.leite@coe.ufl.edu.

BRUCE CARROLL is an Associate Professor in the Department of Mechanical and Aerospace Engineering at the University of Florida. His current research interests center on engineering education with emphasis on peer mentoring and advising processes. His email address is bfc@ufl.edu.

ASHISH AGGARWAL is an Instructional Associate Professor of Computer Science in the Department of Engineering Education at the University of Florida. His research focuses on advancing programming education by developing novel educational technologies and instructional strategies. His email address is ashishjuit@ufl.edu.