

Usage of Conversational Chatbots in Undergraduate Programming courses

-Nandini Gandhi

Flame Scholars Program

Final Project Report

Faculty Mentors: Kaushik Gopalan, Prajish Prasad

Date: 20th March 2024

Executive Summary

The integration of advanced artificial intelligence and generative AI technologies can enhance learning experiences through interactive and student centered approaches. This paper investigates the application of Conversational Chatbots, specifically powered by OpenAI's ChatGPT, in undergraduate programming courses, highlighting the potential of such technologies in education. The first part of the study involves the deployment of a ChatGPT powered chatbot in an 'Introduction to Programming' course which is designed to assist students by providing personalized support and immediate feedback on a range of topics from course logistics to complex programming challenges. This reflects the growing demand for educational tools that are informative, adaptable and capable of engaging students in a more interactive learning process. Further, this study is extended to an advanced programming course, through an analysis of 50+ ChatGPT conversations in a second year undergraduate class. This analysis aims to explore how conversational AI can supplement traditional learning methods by investigating student engagement, the nature of inquiries made, and the chatbot's effectiveness in addressing specific educational needs. By categorizing the types of questions asked and evaluating the chatbot's responses, we seek to identify patterns and trends that could inform the development of more effective AI based educational tools in the future. Ultimately, this research contributes to the broader discourse on the integration of AI in education, highlighting the potential benefits and challenges of conversational chatbots in enhancing the learning experience, facilitating personalized learning, and potentially transforming traditional educational models.

Keywords: Conversational Chatbots, AI Assisted Learning Tools, ChatGPT Implementation, Personalized Learning, AI in Programming Courses, Student Engagement

Introduction

Advanced Artificial Intelligence and Generative AI now has enormous potential for changing the landscape of education, and enhancing learning experiences. This paper focuses on a particular application of AI: the usage of Conversational Chatbots in Undergraduate Programming courses. A chatbot powered by OpenAI's ChatGPT was designed to assist students in an 'Introduction to Programming' course. The project was driven by the growing need for educational tools that are informative, adaptable and interactive, while being capable of providing personalized support to students. The significance of integrating such AI powered chatbots into educational settings goes beyond mere convenience. It addresses a fundamental shift towards more interactive and student-centered learning environments. Programming education, with its complex concepts and practical problem-solving requirements, presents unique challenges that traditional teaching methods alone may not fully address. Here lies the opportunity for AI to make a significant impact, offering a supplementary means of instruction that is both flexible and responsive to individual learning needs.

The development and deployment of this chatbot involved a series of steps, beginning with the customization of ChatGPT to suit the specific needs of the programming course. This process included training the chatbot on course materials and enabling it to handle a wide range of queries from basic course logistics to complex programming problems. The ultimate goal was to create an AI assistant that could enhance the learning experience by providing immediate, relevant feedback, thus facilitating a deeper understanding of programming concepts. To evaluate the effectiveness of the chatbot, we conducted an analysis of its interactions with students, categorizing the types of questions asked and assessing the chatbot's responses for accuracy and helpfulness. This analysis was crucial in understanding

how such technology can best serve educational purposes, identifying strengths and areas for improvement.

Understanding the broader application and significance of conversational AI in educational settings, particularly in programming education, is essential. After establishing the foundation and initial findings related to the chatbot's deployment in a first year 'Introduction to Programming' course, we decided to investigate the role of conversational chatbots in more advanced courses.

The second part of this paper delves into an empirical analysis of 50 ChatGPT conversations from a second year undergraduate programming course, aiming to uncover how students interact with AI to supplement their learning. Firstly, this provides a direct insight into the student engagement with the chatbot, offering a narrower view of the types of questions asked and the chatbot's ability to address specific educational needs. Second, this analysis seeks to identify patterns and trends in the usage of conversational AI, which can inform the development of more effective AI-assisted educational tools in the future.

By examining these conversations, we aim to assess the chatbot's impact on learning outcomes and student satisfaction. Moreover, this comprehensive analysis contributes to the broader discourse on AI in education, highlighting its potential benefits and challenges. It offers valuable insights into how conversational chatbots can be integrated into educational curriculums to enhance the learning experience, facilitate personalized learning, and potentially transform the traditional educational model.

Literature Review

Large Language Models such as ChatGPT have the capability to redefine the way of teaching in our universities through advanced machine learning algorithms and extensive datasets. These models are now being explored by students and educators worldwide for their ability to create intuitive interfaces and changing the way we interact with technology in learning environments. Multiple studies have been conducted on the use of ChatGPT among students, such as the analysis of Twitter information to understand public sentiment on ChatGPT's role in education by Futterer et al. (Fütterer, et al., 2023), and Boubker speaks about the perceived intelligence and system quality of ChatGPT amongst university students, highlighting their trust in using it as a learning aid. (Boubker, 2024)

Pongsakorn Limna et al. investigated the thoughts of professors and students on ChatGPT's role in education. Ten educators and fifteen students were selected from different universities and institutions in Thailand and qualitative research was conducted. The results of this study portrayed a positive perception of ChatGPT as a tool to be used in institutes. Students commended it for providing immediate feedback and answering queries, and educators stated that it reduced their workload by taking over routine tasks and allowing them to concentrate on other complex educational activities. However, the study also highlighted concerns about the accuracy of the chatbot's information. The information generated by GPT depended heavily on the quality of the input data and hence, the design of the prompt given to ChatGPT. (Pongsakorn Limna et al., 2023)

Elnashar, Ashraf, et al. highlight the importance of prompt engineering in order to make full use of the capabilities of large language models like ChatGPT in coding education. The study showed that ChatGPT can generate code solutions which are competitive to the top human

written solutions available on Stack Overflow, given that the prompt is well crafted. The study was conducted by generating 100 different solutions to coding problems and selecting the highest quality output, which was then compared to the top voted solution by Stack Overflow. The ChatGPT solution performed equally, and at times better than the human solution. It was observed that the best solution was produced by the most effective prompt which guided the model through the problem solving process step by step. The results highlighted the importance of prompt engineering to maximize the potential of large language models like ChatGPT in coding educational settings. (Elnashar, Ashraf, et al.)

Recognizing the importance of ChatGPT in the educational process, Lin speaks about the integration of ChatGPT within Moodle which is a widely used Learning Management System (LMS). The aim of this paper is to use the capabilities of ChatGPT and harness the potential of AI to aid education by contextual conversing, feedback response, adaptive tutoring, analytical power and multilingual capabilities. This paper integrates ChatGPT into Moodle using a three step technical procedure. The beneficial capabilities of this model include numerical calculations, data generation and interpretation, course design, extended learning and emotional support. The paper lays the foundational basis of the model, upon which there are a multitude of building capabilities that stakeholders of higher education could adopt. This includes its ability to act as a co-pilot in educational tasks like scientific writing and data analysis. The paper acknowledges the few drawbacks to the integration of AI in education, such as, copyright issues, over dependency, need for new assessment methods, feedback accuracy, data security and privacy. However, there are also a range of areas where its importance is undeniable- such as, providing a road map for educators to enhance teaching and learning experiences. (Lin, 2023)

It is evident that the application of Large Language Models (LLMs) like ChatGPT in educational settings is broadening. The study by Liffiton et al. introduces CodeHelp, a tool designed with ‘guardrails’ to help programming students without directly revealing solutions, emphasizing the balance between providing aid and encouraging independent problem solving skills. (Mark Liffiton et al., n.d.) Similarly, another study presents CodeAid, an LLM-based programming assistant that focuses on delivering technically correct responses, which includes pseudo code and annotations for student code without giving direct code solutions. (Majeed Kazemitabaar et al., n.d.). This aims to enhance conceptual understanding and debugging skills among students. These additions highlight the ongoing study of LLMs in education, and in programming classes where the balance between offering support and promoting learning is important. The development of these tools reflects an effort to make use of the capabilities of LLMs while being aware of their potential to cause over reliance or diminish learning outcomes.

While research demonstrates the potential of ChatGPT and similar technologies to support and enhance educational models, there remains a relatively unexplored gap in customizing these technologies for undergraduate courses. This paper seeks to fill that gap, detailing the implementation of a ChatGPT-based chatbot in an ‘Introduction to Programming’ course and further analyzing its application in more advanced programming courses. By examining ChatGPT’s effectiveness at different levels of programming education, this study provides an overview of the tool’s potential to support student learning and contribute to the evolution of educational models.

Methodology

The chatbot was used in the “Introduction to Programming” class taken by approximately 70 students over a period of 2 months. The chatbot, as seen in Figure 1, has a simple user interface. When a user starts a conversation, the chatbot already has a predefined system prompt set up in the backend. The system prompt was trained on specific information about the course such as the professor’s name, class timings, exam schedules, and syllabus, among others. It was also trained on which topics questions should be answered, and which topics to avoid.

Example of sections of the prompts:

You are the course assistant for the CSIT101: Introduction to Programming course. The professor for this course in Term 2 is Prof. Kaushik Gopalan.

If a user asks about advanced topics not in the syllabus like OOP or generators please inform them that it is not in the syllabus.

This system prompt, trained with specific instructions and context on how to answer the question, is sent to the Large Language Model (LLM), which is OpenAI’s ChatGPT, at the start of every conversation. When a user types a question, the API - a tool that allows the backend to interface with OpenAI’s servers, where the ChatGPT model is hosted - directs the question to the LLM. To be able to understand the context and provide a relevant response, the LLM uses the system prompt as well as the user input. The response is generated by the model based on its extensive training and the information provided in the system prompt. After the LLM produces a response, the chatbot’s interface receives it via the API. The user can see the response and interact further by asking new questions or more information. Throughout every individual conversation, the system prompt remains in the background,

providing context for the LLM and ensuring a relevant conversation. As seen in the block diagram, this process ensures that every user question is answered with respect to the system prompt enabling the chatbot to act as a personalized learning assistant.

Prompt Segmentation and Keyword Detection

The main challenge we face is the vastness of the system prompt in comparison to the specific nature of students' questions. These questions focus on particular topics, such as details about the course or programming issues, which require only a small part of the information in the system prompt for accurate responses. Hence, we break down the system prompt into smaller, focused sections that cater to different types of questions. This breaking down is done using a keyword detection algorithm that identifies key terms in the students' queries, directing the system to the most relevant section of the prompt for generating responses. This method of segmentation is important because the entire system prompt can be very large, potentially with thousands of words. To put it into perspective, if the system prompt is around 2000 words, this could translate to approximately 4,000 tokens. Hence, it is not practical to process the entire system prompt for every query. Instead, by identifying key terms and matching them with the corresponding segment of the prompt, the chatbot can more efficiently generate relevant responses without the need to go through unnecessary information.

For example, if the question included the word “professor”, it would use the system prompt segment related to the course logistics. After the keyword has been selected, the chatbot, through the system, would select the appropriate prompt which would then allow the LLM to generate an appropriate response. Figure 2 illustrates the entire process, including how the appropriate system prompt is selected. When the user inputs a question, a keyword search is

triggered. If a keyword is matched, it goes to the prompt segment selected before the query reaches the LLM. The model then sends this segmented prompt to the LLM along with the query. Using this, the LLM generates a response which is fetched and posted to the user.

Moving from the basics to a more advanced level, we took a closer look at how second-year students in a programming course were interacting with a chatbot, and in this case, with ChatGPT. The Introduction to C++ class had 78 students in 2 sections (41 + 37), and the course ran from Sept 2023 to Dec 2023. Their final project included topics such as “Simulate an app which you use – e.g. Amazon, Instagram, Ola, Zomato”, or “Use existing libraries to build an application – e.g. opencv, csound, OpenAL”, to be done in groups of 4-5. The professor asked these groups to keep a record of their conversations with ChatGPT while working on their final C++ project. This led to the collection of 50 conversations and a total of 389 questions posed to ChatGPT, which were analyzed to understand how students interact with AI to aid their learning in programming.

Analysis and Results

Part A: Analysis of the chatbot used in a beginner programming class

After the chatbot was created, it was used by the students of the ‘Introduction to Programming’ class. We categorized student questions into distinct types:

1. Course Logistics: For questions about the class administration, such as, ‘What are Prof. Kaushik’s office hours?’ or ‘When is the midterm exam?’, the chatbot provided essential logistical information.
2. Programming Concepts: For conceptual questions such as, ‘Can you explain how Python lists work?’ or ‘What is the difference between a for loop and a while loop?’, the chatbot provided explanations to clarify these programming concepts.

3. Practice Problems: When asked for practical practice questions such as these: ‘Could you give me a practice problem on string manipulation?’ or ‘Generate a question involving dictionaries.’, the chatbot generated practice questions within the syllabus of the course.

As seen in Figure 3 and Table 1, there were 209 individual questions asked in total- out of which, Course logistics made up 37.3% of the questions (78 in total), Programming Concept questions made up 52.6% with 110 questions, and Practice Problems questions were 10% of the total (21 Questions). There were approximately 95 conversations that encompassed these questions, with an average of 2 questions per conversation.

The chatbot accurately provided information about the CSIT101 course. For example, it correctly identified that the professor for Term 2 is Prof. Kaushik Gopalan, and stated that the class is held in room APJ001. The chatbot also correctly answered questions about the class schedule, stating that classes are held on Tuesdays and Thursdays from 2:15 pm to 4:10 pm.

The chatbot answers questions related to the course syllabus with a high accuracy. It covered the range of Python topics in the syllabus- including variables, data types, string operations, data structures (lists, dictionaries), control flow (conditional statements, loops), functions, file I/O, data cleaning, data manipulation, and basic data visualization. It also consistently refused to respond to questions that are unrelated to the CSIT101 course or the Python programming language. For example, when asked the height of the Eiffel Tower, it refused to answer the question with the justification that it was unrelated to the course content.

However, while handling some of the code related questions, the chatbot sometimes failed. Despite the explicit instruction in the prompt to not answer questions about advanced topics

such as classes and Object-Oriented Programming (OOP) that are not in the syllabus, the chatbot sometimes provided responses on concepts like inheritance. It refused to answer questions that directly say the word “OOP”, because the prompt asks not to delve into OOP, but it will answer questions on topics which fall under the domain of object oriented programming. For instance, it can sometimes incorporate classes in its code or answer questions pertaining to the syntax required to implement classes in Python.

The development of this AI powered programming chatbot provides students with a learning tool that can assist them outside of the structured classroom hours. It reduces the need for office hours, hence, saving time and effort for professors. Through ongoing refinement and prompt engineering, this chatbot can be adapted to courses across many disciplines, such as Psychology, Finance, and Economics, among others, and act as a supplemental learning tool.

Part B: Analysis of the use of ChatGPT in an intermediate programming class

As we moved our focus to the second year of undergraduate programming, we focused on the application of ChatGPT in an intermediate programming class, where students typically encounter a broader and more complex range of problems. The methodology adopted for analyzing these conversations involved categorizing each question-answer pair into specific areas of inquiry. These areas included Modifying Code, Generate Code, Explain Code, Configuration Questions, Syntax Debugging, Output Debugging, and Non-Programming questions.

Modifying Code involves requests to adjust existing code, enhancing functionality or rectifying errors. For example: “Can you please edit this code so that the input is in the 24 hour clock in the format 1300 instead of 1:00 PM and an appropriate example is given.”

Generate Code refers to seeking new code examples to understand the application of concepts, a direct approach to learning through example. For example: “Show how to write a C++ function for calculating factorials.” **Explain Code** captures the need for clarifications on the workings and logic behind specific code segments, essential for deepening coding comprehension. For example: “Explain the operation of this code snippet.” **Configuration Questions** deal with the setup and troubleshooting of programming environments, a foundational step for practical coding exercises. For example: “How do I open an image file in C++?”. **Syntax Debugging** focuses on identifying and correcting code that doesn't execute due to syntax errors, a crucial skill for any programmer. Example: “Resolve a syntax error in this if-statement.” **Output Debugging** addresses the alignment of a program's actual output with its expected outcome, fostering problem-solving abilities. Example: “Fix an issue where a sum operation yields incorrect output.” Lastly, **Non-Programming Questions** encompass all inquiries outside the technical scope, covering course content, administrative details, or general learning advice. This classification helps to identify the diverse needs of students while interacting with the chatbot.

To streamline the analysis, these categories were further grouped into three main types: Debugging, Coding, and Conceptual. Debugging covered Syntax and Output Debugging, which focuses on diagnosing and fixing errors in code. Coding included ‘Generate Code’ and ‘Modifying Code’, which deals with creating new code snippets or altering existing ones. Conceptual Questions included ‘Explain Code’ and ‘Configuration Questions’, aimed at understanding programming concepts or how to set up programming environments.

The conversations that contained more than seven question-answer pairs were given special attention to identify the dominant category of the chat within these extended interactions.

This approach allowed for a deeper understanding of which aspects of programming education students are most engaged with and where they seek the most assistance from AI. By categorizing the content of the conversations in this way, it is possible to get a clearer picture of how tools like ChatGPT can be optimized to better serve educational purposes in programming courses.

To begin with, we analyzed the types of questions students asked the chatbot, which, as seen in Figure 4, showed that the majority of questions were about ‘Syntax Debugging’ with 108 questions, making up 27.8% of the total. This was followed by ‘Modifying Code’ with 87 questions and ‘Generate Code’ with 53 questions, indicating that students were actively engaging in writing and improving their programming. ‘Configuration Questions’ and ‘Explain Code’ also featured prominently, pointing to a need for understanding programming environments and the logic behind code. The least amount of queries were ‘Non-Programming’ related with 23 questions, yet these are important as they reflect the wider context in which students are learning to program.

Following the pie chart analysis, we investigated the nature of student inquiries by examining the longer and more complex dialogues comprising over seven question-answer pairs. This weighted graph, depicted in Figure 5, provides a visual representation of the interconnectedness of various categories of student questions. In this graph, nodes represent distinct categories such as ‘Syntax Debugging’, ‘Generate Code’, and ‘Modifying Code’, among others. Edges connecting these nodes indicate the transition between question types, which correspond to the frequency of questions within that category. This graph paints a picture of the student learning process in action. For instance, the substantial paths between ‘Syntax Debugging’ and ‘Modifying Code’ show a common progression where students first

resolve syntax errors before proceeding to refine their code. This pattern is indicative of the iterative nature of programming, where debugging is not an isolated task but part of a cycle of code development and improvement. Moreover, the connection between ‘Configuration Questions’ and other categories indicates that students are not only learning to write code but are also actively engaged in setting up their programming environments. The relatively smaller, yet significant, node for ‘Explain Code’ points to a crucial aspect of programming education which is the need for conceptual understanding to support practical skills.

This weighted graph serves as a precursor to Table 2, providing a clear distribution across various categories. The analyzed data reveals several patterns in the types of questions students posed and the dominant categories of their inquiries.

Debugging stands out as the dominant category in most of the conversations with ChatGPT, suggesting that students frequently encounter and seek help for issues related to code errors. This is indicative of a natural focus on troubleshooting as a critical component of the programming learning process. Questions related to coding also represent a substantial fraction, highlighting students’ efforts to create and modify code, which is central to their hands-on learning experience. Conceptual questions, while less frequent, are still present, showing that students also use the chatbot to understand the theoretical underpinnings of programming concepts.

Continuing from the analysis of the question-answer pairs and the identification of dominant categories, we have utilized pattern recognition to further our understanding of the specific patterns in student queries.

Starting with the debugging questions, the weighted graph in Figure 6 presents a visual exploration of the Debugging interactions. The prominence of Syntax Debugging with 66

interactions highlights a common challenge students face in writing syntactically correct code. The heavy loopback to itself indicates that students often follow up with additional syntax-related questions, implying a cycle of trial and error as they learn. With 17 interactions in the Output Debugging category, students also frequently encounter discrepancies between expected and actual program outputs. The connections of Output Debugging from and to Syntax Debugging suggest that issues with code functionality are often tied to underlying syntax errors. Modifying Code is an area closely tied to both Syntax and Output Debugging, with 30 interactions. The flow between Modifying Code and the two debugging categories suggests a dynamic where resolving syntax or output issues directly influences the students' ability to modify code effectively. The directional flow between Syntax Debugging and Output Debugging, and between these categories and Modifying Code, indicates a pattern where students might start with a syntax issue, proceed to resolve output problems, and then work on modifying their code to achieve the desired functionality. The loopbacks and the weighted edges show a recursive learning pattern, where resolving one issue often leads to revisiting previous topics or advancing to new ones. This interactivity could be useful for professors to understand because it could affect the development of more targeted instructional strategies using AI tools like ChatGPT.

Moving on to the Coding category, the pattern recognition analysis using the weighted graph in Figure 7 sheds light on student interactions in areas of generating and modifying code. With 29 interactions, the 'Modifying Code' node indicates that students are actively working on adjusting and improving their code. The large loop indicates that students are often revisiting their code modifications, possibly to refine their solutions. "Generate Code" shows 26 interactions, signaling that students are also heavily engaged in creating new code segments. The transitions between Modifying Code and Generate Code suggest a back and forth process as students iterate over their coding tasks. Although less connected in this graph

compared to the Debugging category, Syntax Debugging still plays a role, with 7 interactions. This implies that while writing or modifying code, students occasionally run into syntax issues that need clarification. The graph also indicates that configuration settings are a relevant aspect for students when they are setting up their coding environment or preparing to write code.

The directional flow between nodes, especially the heavier weights from Modifying Code to Generate Code (and vice versa), points to a cycle of code creation and refinement. Students are not only writing new code but also revisiting and tweaking it, which is an essential part of learning to code effectively. This pattern analysis from the Coding category reinforces the idea that learning to program is a highly iterative process, involving constant refinement and troubleshooting. Understanding these patterns can help professors and AI developers to tailor the chatbot's responses better, aligning with the students' learning workflows.

For the Conceptual category, the pattern recognition graph in Figure 8 illustrates student interactions around understanding the broader concepts of programming and configuration. With 15 interactions and a significant self loop, the "Configuration Questions" node suggests that students often revisit configuration issues, reflecting a continuous need to understand the setup and environment in which they are programming. The reciprocal link between Configuration Questions and Explain Code suggests that students who are configuring their environments may also be looking to understand how their setup affects the code they write or vice versa. This pattern demonstrates that students are not only dealing with immediate coding challenges but also with the foundational elements that will inform their long-term programming skills.

The deployment of ChatGPT within the C++ class shows us how students interact with technology to enhance learning. The analysis of student questions across Debugging, Coding,

and Conceptual categories highlights an interaction with the chatbot which is primarily centered on practical problem solving and understanding programming concepts.

Conclusion

Throughout this analysis into the use of conversational chatbots in undergraduate programming courses, we have seen that tools such as ChatGPT can significantly enhance the way students learn and interact with course material. The practical application of this AI driven technology in an 'Introduction to Programming' course has demonstrated its effectiveness in providing students with tailored support. However, the chatbot sometimes struggled with questions outside the set syllabus or advanced topics it wasn't programmed to handle. This limitation suggests that there is room to enhance ChatGPT's knowledge and its ability to manage more complex or unrelated questions, which could make it more valuable as an educational resource. The use of ChatGPT covered important areas such as course logistics, understanding programming concepts, and offering practice problems. These aspects highlight how ChatGPT can support students in various ways, from giving easy access to course information, deepening their understanding of programming, to improving their problem-solving skills. This versatility shows the chatbot's potential to aid learning in diverse ways and how it could change teaching methods by providing a more personalized and responsive learning experience.

The extension of our study to an intermediate programming class has further solidified the chatbot's role as an invaluable educational resource. This segment of the study provided insights about the interaction dynamics between students and the AI tool. We categorized student queries into several main areas, including debugging, coding, and conceptual understanding, which allowed us to dissect the nature and scope of questions asked to

ChatGPT. ChatGPT offered personalized guidance and allowed students to navigate through the complexities of learning a programming language. It functioned effectively as an on-demand teaching assistant, addressing a variety of questions that students encountered during their course work.

In summary, this paper demonstrates the potential of AI assisted learning tools to supplement traditional educational methods. ChatGPT facilitated a responsive learning environment where students could interact about their programming challenges. While there are challenges to perfecting the integration of AI into learning environments, there is potential for a positive impact on student learning and a reduction in professors' administrative burdens.

There are several avenues to incorporate AI tools in the classroom, especially for programming courses. One interesting application would be for the bot to prepare mock question papers similar in style to previous samples from the instructor of the course. Additionally, AI tools that generate code snippets with specific key lines of code left blank for the student to fill in are likely to be a valuable in-class learning resource. The next few years will undoubtedly see an explosion in AI tools that enhance the classroom experience for students in ways that will make the initial experiments detailed in this manuscript seem insignificant; we look forward in anticipation to whatever surprises this AI-aided future may bring.


References

- Boubker, O. (2024). From chatting to self-educating: Can AI tools boost student learning outcomes? *Expert Systems with Applications*, 238 (121820), 121820.
- Elnashar, A. (n.d.). *Prompt Engineering of ChatGPT to Improve Generated Code & Runtime Performance Compared with the Top-Voted Human Solutions*.
- Lin, J. (2023). *ChatGPT and Moodle Walk into a Bar: A Demonstration of AI's Mind-blowing Impact on E-Learning*. *SSRN Electronic Journal*.
- Fütterer, T., Fischer, C., Alekseeva, A., Chen, X., Tate, T., Warschauer, M., & Gerjets, P. (2023). ChatGPT in education: Global reactions to AI innovations. *Scientific Reports*, 13(1). <https://doi.org/10.1038/s41598-023-42227-6>
- Majeed Kazemitabaar, Runlong Ye, Xiaoning Wang, Austin Z. Henley, Paul Denny, Michelle Craig, & Tovi Grossman. (n.d.). *CodeAid: Evaluating a classroom deployment of an LLM-based programming assistant that balances student and educator needs*. <https://arxiv.org/html/2401.11314v1>
- Mark Liffiton, Brad Sheese, Jaromir Savelka, & Paul Denny. (n.d.). *CodeHelp: Using large language models with Guardrails for scalable support in programming classes*. <https://arxiv.org/abs/2308.06921>
- Pongsakorn Limna, Tanpat Kraiwanit, Kris Jangjarat, Prapasiri Klayklung, & Piyawatjana Chocksathaporn. (2023). The use of ChatGPT in the Digital Era: Perspectives on chatbot implementation. *I*, 6(1). <https://doi.org/10.37074/jalt.2023.6.1.32>

Tables and Figures

Welcome to Pyndora's bot: Intro Python@FLAME

Hello, I am the course assistant for the CSIT101: Introduction to Programming course. You can ask me any questions you have about the course, or about the course material. I will help you to the best of my ability. I am powered by chatGPT, so do forgive the occasional blooper...



Type your question here...

Ask
Sample Questions

The use of this chatbot is completely voluntary. The questions and responses are logged, but your identity is not saved anywhere.

Figure 1. Simple user interface of the ChatGPT-powered chatbot used in the "Introduction to Programming" course

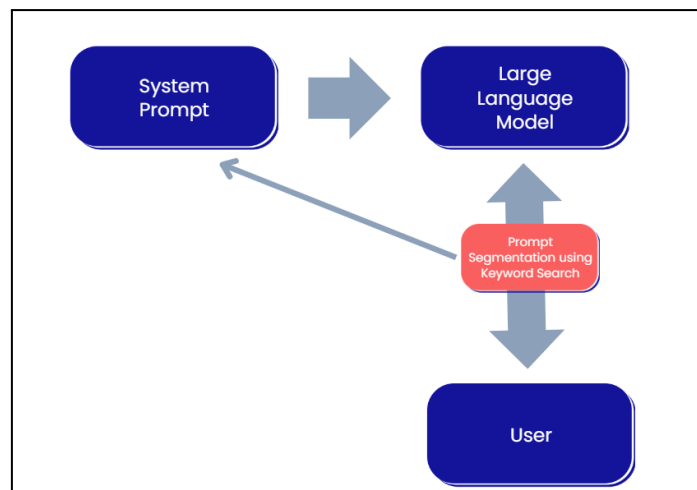


Figure 2. Backend process of the Chatbot with Prompt Segmentation using Keyword Search

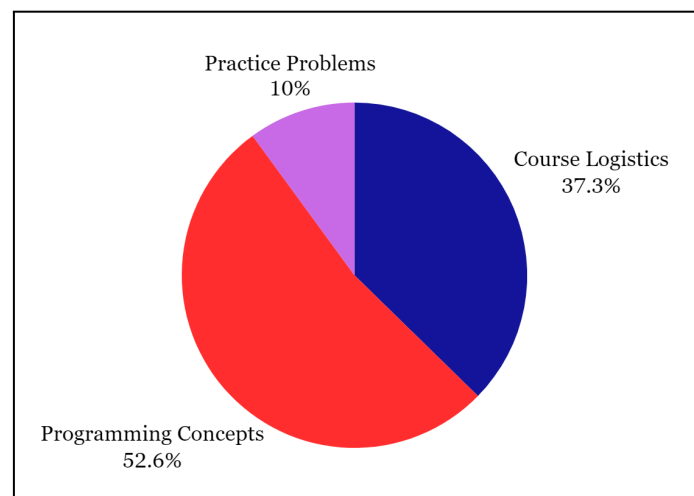


Figure 3. Distribution of questions across different types

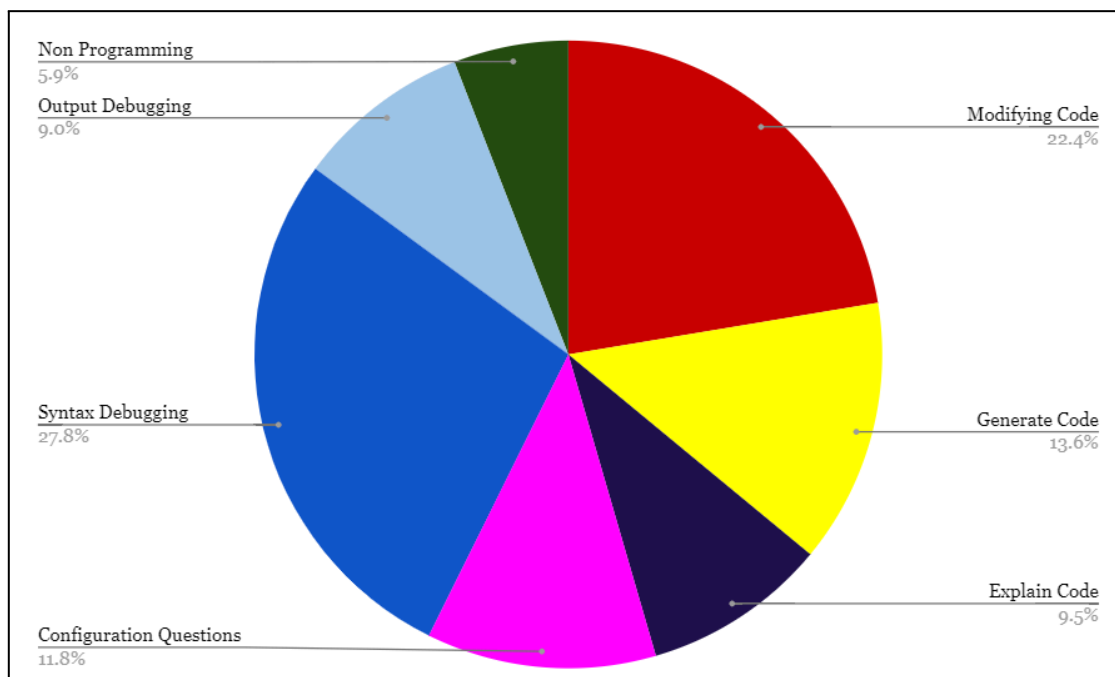


Figure 4. Question-Answer pairs categorized into types

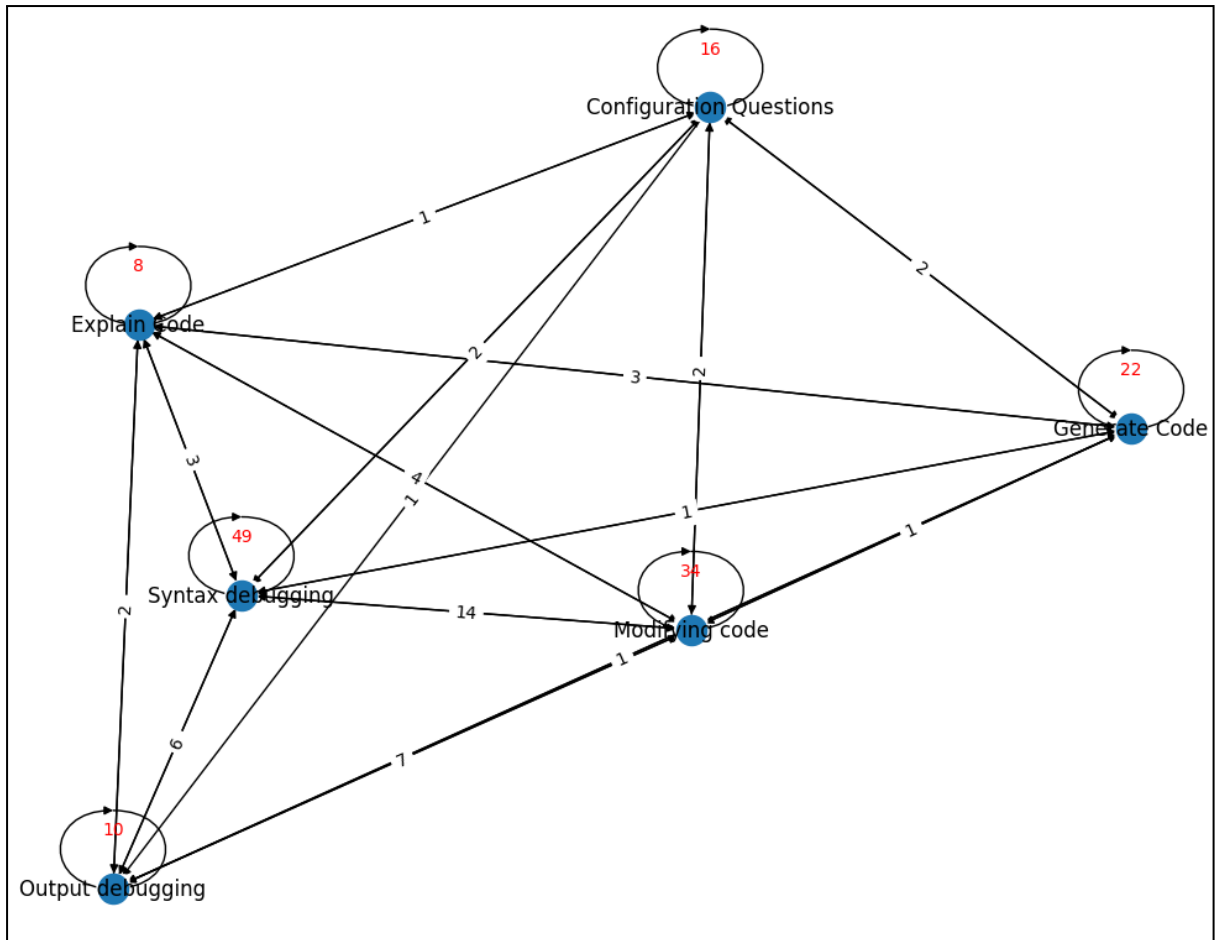


Figure 5. Relationship and frequency of transitions between different categories of student questions, illustrating the learning process.

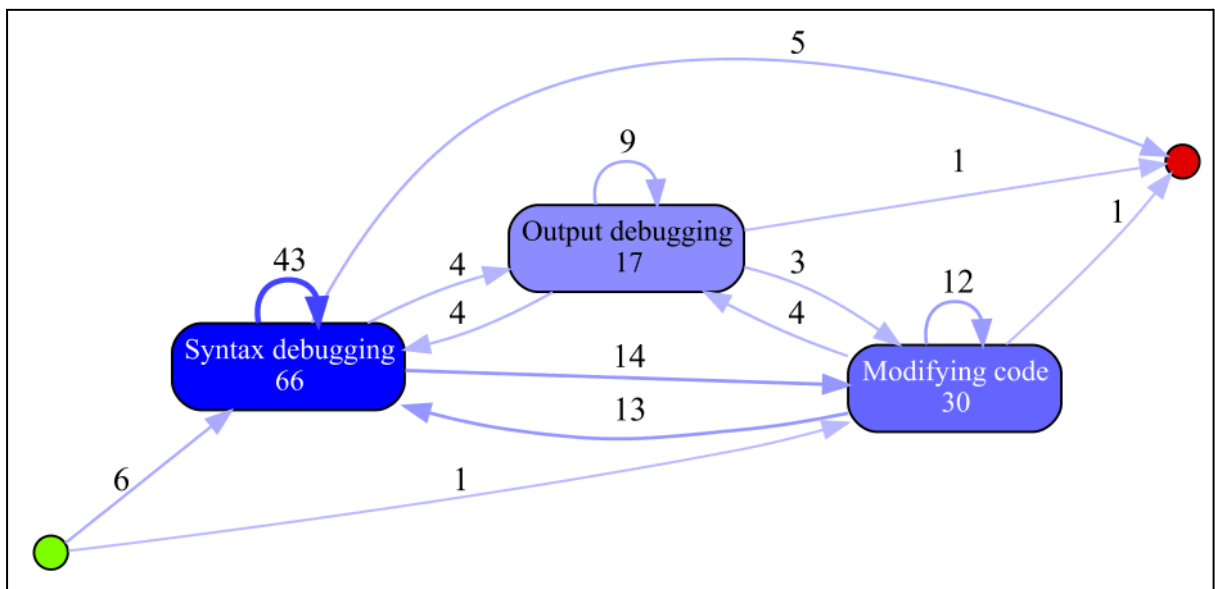


Figure 6. Weighted graph generated from 'Debugging' Questions

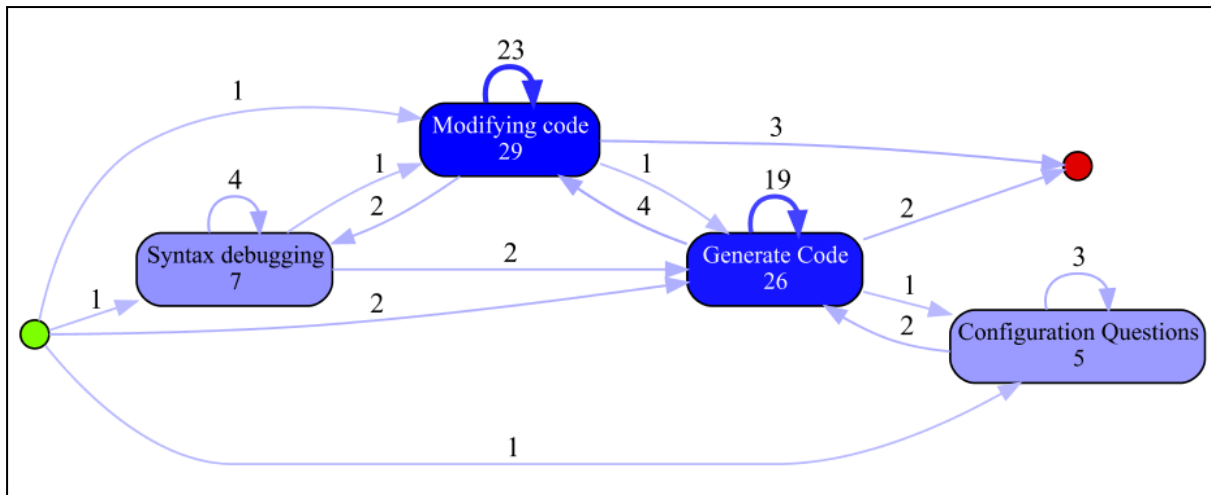


Figure 7. Weighted graph generated from ‘Coding’ Questions

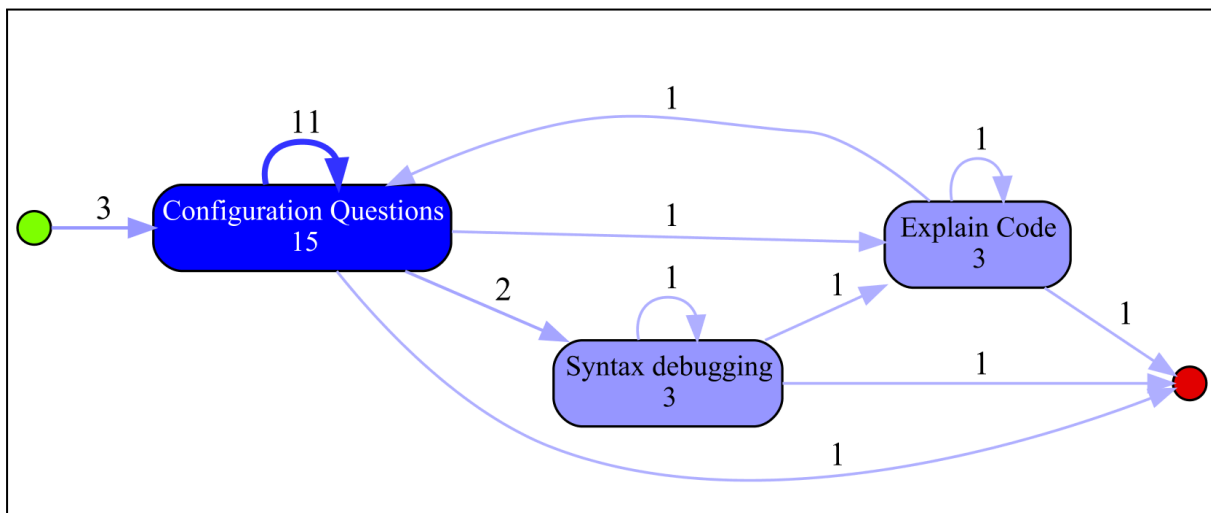


Figure 8. Weighted graph generated from ‘Conceptual’ Questions

| Question Types | Number of Questions |
|----------------------|---------------------|
| Course Logistics | 78 |
| Programming Concepts | 110 |
| Practice Problems | 21 |
| Total | 209 |

Table 1. Distribution of questions across different types

| Conversation Name | Dominant Category | Debugging Proportion | Coding Proportion | Conceptual Proportion |
|-------------------|-------------------|----------------------|-------------------|-----------------------|
| Sheet 1 | Debugging | 51.85 | 37.04 | 11.11 |
| Sheet 2 | Debugging | 50 | 45.83 | 4.17 |
| Sheet 3 | Debugging | 50 | 50 | 0 |
| Sheet 4 | Debugging | 53.85 | 38.46 | 7.69 |
| Sheet 5 | Conceptual | 37.5 | 0 | 62.5 |
| Sheet 6 | Mixed | 37.5 | 31.25 | 31.25 |
| Sheet 7 | Conceptual | 0 | 14.29 | 85.71 |
| Sheet 8 | Coding | 0 | 87.5 | 12.5 |
| Sheet 9 | Coding | 0 | 93.75 | 6.25 |
| Sheet 10 | Debugging | 81.82 | 18.18 | 0 |
| Sheet 11 | Debugging | 77.78 | 7.41 | 14.81 |
| Sheet 12 | Coding | 33.33 | 66.67 | 0 |
| Sheet 13 | Mixed | 16.67 | 44.44 | 38.89 |
| Sheet 14 | Coding | 15.79 | 52.63 | 31.58 |
| Sheet 15 | Debugging | 100 | 0 | 0 |
| Sheet 16 | Coding | 31.25 | 62.5 | 6.25 |
| Sheet 17 | Conceptual | 12.5 | 0 | 87.5 |

Table 2. Patterns in the questions students posed and dominant categories of their inquiries