



”Like a Nesting Doll”: Analyzing Recursion Analogies Generated by CS Students Using Large Language Models

Seth Bernstein, Paul Denny, Juho Leinonen, Lauren Kan,
Arto Hellas, Matt Littlefield, Sami Sarsa and Stephen MacNeil

EasyChair preprints are intended for rapid dissemination of research results and are integrated with the rest of EasyChair.

March 21, 2024

“Like a Nesting Doll”: Analyzing Recursion Analogies Generated by CS Students using Large Language Models

Seth Bernstein
Temple University
Philadelphia, PA, United States
seth.bernstein@temple.edu

Paul Denny
University of Auckland
Auckland, New Zealand
paul@cs.auckland.ac.nz

Juho Leinonen
University of Auckland
Auckland, New Zealand
juho.leinonen@auckland.ac.nz

Lauren Kan
Temple University
Philadelphia, PA, US
lauren.kan@temple.edu

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

Matt Littlefield
Temple University
Philadelphia, PA, US
matt.littlefield@temple.edu

Sami Sarsa
Aalto University
Espoo, Finland
sami.sarsa@aalto.fi

Stephen MacNeil
Temple University
Philadelphia, PA, US
stephen.macneil@temple.edu

ABSTRACT

Grasping complex computing concepts often poses a challenge for students who struggle to anchor these new ideas to familiar experiences and understandings. To help with this, a good analogy can bridge the gap between unfamiliar concepts and familiar ones, providing an engaging way to aid understanding. However, creating effective educational analogies is difficult even for experienced instructors. We investigate to what extent large language models (LLMs), specifically ChatGPT, can provide access to personally relevant analogies on demand. Focusing on recursion, a challenging threshold concept, we analyzed the analogies generated by 385 first-year computing students. They were provided with a code snippet and tasked to generate their own recursion-based analogies using ChatGPT, optionally including personally relevant topics in their prompts. We observed a great deal of diversity in the analogies produced with student-prescribed topics, in contrast to the otherwise generic analogies, highlighting the value of student creativity when working with LLMs. Not only did students enjoy the activity and report an improved understanding of recursion, but they described more easily remembering analogies that were personally and culturally relevant.

CCS CONCEPTS

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

KEYWORDS

analogies, large language models, computing education

ACM Reference Format:

Seth Bernstein, Paul Denny, Juho Leinonen, Lauren Kan, Arto Hellas, Matt Littlefield, Sami Sarsa, and Stephen MacNeil. . “Like a Nesting Doll”: Analyzing Recursion Analogies Generated by CS Students using Large Language Models. In *Proceedings of Proceedings of the 2024 ACM Conference*

on *Innovation and Technology in Computer Science Education (ITiCSE '24)*. ACM, New York, NY, USA, 7 pages.

1 INTRODUCTION

Learning to program is challenging because students may struggle to connect unfamiliar terms and concepts to existing knowledge and to their everyday lived experiences. Especially for difficult threshold concepts [4, 37], like recursion, students benefit from high-quality explanations or visual representations such as visualizations and concept maps [37]. Analogies and metaphors are another useful pedagogical technique which connect unfamiliar concepts to ideas that students already intuitively understand [8, 14, 21, 39]. Analogies and metaphors have been previously studied in computing education contexts [1–3, 7, 11, 13, 16, 24]. However, it has been shown that students struggle to craft high-quality analogies and that they could benefit from additional scaffolding [17].

Large language models (LLMs) have recently been shown to generate high-quality explanations [25, 28]. Students even prefer explanations generated by an LLM to ones generated by their peers [25]. Consequently, LLMs may be useful tools for scaffolding students trying to craft analogies. However, although there is some evidence that LLMs can generate analogies [9, 30], there has not yet been any systematic investigation into how students might use LLMs to generate their own analogies. Given this gap in the literature, we investigated the following research questions:

RQ1 What types of analogies do students generate to explain recursion when given access to an LLM?

RQ2 What do students think about using LLMs to generate their own analogies?

In this paper, we conducted a study with a sample size of over 350 students at a large public research university. As part of their lab on recursion, a well-known threshold concept that is challenging for many students [4, 37], students were asked to use ChatGPT to craft their own analogies to explain recursion using any personally relevant topic as a theme. We collected student responses to survey

questions and analyzed their generated analogies and corresponding prompts. Our results show that when students explicitly specified the topic of the analogy in their prompt, the resulting analogies were more diverse, featuring a different distribution of topics compared to those produced by the LLM when topics were not prescribed. This suggests that involving students in crafting analogies with ChatGPT leads to more creative and relevant analogies compared to relying solely on the model. Students expressed positive sentiments about the activity, noting instances where the model-generated analogies aided their understanding and retention. Previous studies have established the effectiveness of analogies in computing education [1–3, 7, 13, 16], which is not the focus of this paper. Instead, this study focuses on what analogies students produce and their potential for engagement. Future work will focus on systematically investigating the impact on student learning.

2 RELATED WORK

2.1 Analogies to Explain Computing Concepts

Explanations are critical for helping students understand complex topics. Over two decades of computing education research has demonstrated how practicing ‘explain in plain English’ questions [43] and self-explanations [33, 41, 42] can provide students with short-term and long-term benefits. More recently, computing education researchers have begun investigating the use of analogies to make unfamiliar concepts more approachable for students. Often, analogies are carefully developed by instructors to convey complex concepts such as parallel programming [2, 3, 13], introductory computing [7, 17, 19], and algorithms and data structures [11].

Analogies are a pedagogical tool that equates two disparate concepts based on their similarities in at least one aspect. In computing education, where many unfamiliar terms and concepts may seem completely disconnected from previously learned concepts and lived experiences, analogies have the ability to connect these complex concepts to ones that are more familiar to the learner. For example, an algorithm is like a recipe, a bitwise operator is like a light switch, and a memory address is like a house number. Analogies do not need to perfectly describe the concept to be useful. Analogies help to provide an intuition for the concept which may help students get passed an initial hurdle in understanding. In recent work, Harper et al. investigated the potential of student-generated analogies in computing education, particularly in fostering comprehension of abstract computing concepts like variable scope [17]. Their work showed that students faced challenges in achieving complete and precise structural alignment, along with maintaining an appropriate level of abstraction across their analogy mappings. Bettin et al. showed that when providing students with analogies, preferences about the analogies varied across students and students did not always use a given analogy in the same way [2, 3]. These prior studies suggest that students benefit from analogies but that analogies are ideally personalized based on each student's needs and experiences. Additionally, students could greatly benefit from targeted scaffolding to address these challenges and enhance their ability to generate effective analogies.

2.2 Generating Learning Materials with LLMs

Large language models (LLMs) provide opportunities for educators to generate many high-quality learning materials at scale [10, 25, 28–30]. Finnie-Ansley et al. demonstrated that Codex could produce programming assignments that were useful as a first draft for educators [10]. Since then, LLMs have been used to generate explanations [25–28, 30], programming assignments [38, 40], and to respond to students’ help requests [20, 22]. More recent work has provided empirical evidence that students are legitimately relying on Generative AI as a source of help when they get stuck [22, 34, 44].

Several papers have recently explored the potential of LLMs to enhance code comprehension, with a particular focus on their ability to explain code to students. In an initial study by MacNeil et al., the researchers demonstrated that LLMs were able to trace code execution, articulate the high-level purpose, and offer analogies for programming concepts [30]. This initial investigation focused on the capabilities but not on the associated quality of the explanations and analogies. MacNeil et al. later extended their investigating by studying students’ preferences and engagement with explanations generated by LLMs, considering variations in the types of explanation provided [28]. Leinonen et al. demonstrated that the explanations produced by LLMs were preferred by students when compared to explanations generated by their peers [25]. Despite the considerable attention LLM-generated explanations have received, there remains a gap in the literature regarding a systematic study of analogies generated by students with the assistance of LLMs.

3 METHOD

3.1 Study Context

The data for this study were collected from an introductory programming course taught in Spring 2023 at the University of Auckland, a large public research University in New Zealand. The 12-week course used C as the programming language and is a required course for students in the first year of the Engineering program. These students specialize in their second year based on GPA, and thus students in the course have an incentive to perform well in all of their first-year courses, even if they do not intend to specialize in computing topics.

When our data were collected, 889 students were enrolled in the course and we collected 841 responses (95% of enrolled students). The course featured weekly lab sessions that generally included programming exercises that are automatically graded. Ethics approval was granted by the institution (UAHPEC25279).

3.1.1 Recursion: The topic of recursion is introduced in the penultimate week of the course, and is the focus of the final lab in Week 12. As a threshold concept [37], recursion is considered a difficult topic for many students. This is supported by the literature [15, 35], as well as comments submitted by students on the end-of-semester teaching evaluations (examples from 2022 included: “*spend some more time on recursion because I feel like it’s quite a difficult topic to get on top of*” and “*I found the visual representation of recursion and memory quite hard to get my head around*”). Furthermore, understanding recursion goes beyond memorizing definitions; it encompasses the ability to apply, test, and utilize the concept in real-world scenarios, thereby opening up new avenues for mental models and program design. The intricacies of recursion include both

theoretical significance and substantial practical application, making it a transformative element of computing education [23].

3.2 Materials

In addition to the standard lecture materials used in the course for teaching recursion, which included recursive images (i.e. the Droste effect¹), visualizations of stack frames during recursive calls, and common but simplistic text-book examples (such as factorial and Fibonacci), students were also provided with an example of an analogy as part of the preparation materials for the Week 12 lab. This example was prefaced by the statement: *"Analogies can be useful for teaching difficult concepts because they relate unfamiliar ideas to something already known, making the new information easier to understand. For instance, an analogy often used in physics textbooks compares electrical circuits to hydraulic circuits"*, and then demonstrated the output from ChatGPT when asked to *"use an analogy to explain how electrical circuits work"*. The output compared electrical circuits to a plumbing system, drawing comparisons between voltage/pressure, current/flow rate, and resistance/constriction.

The first exercise in the lab asked students to generate an analogy using ChatGPT to explain a recursive function that appeared in the course textbook. The function prints a string in reverse, and is shown in Figure 1. The instructions given to students for this exercise are shown in Figure 2. We assumed that many (but not all) students had OpenAI accounts for ChatGPT access. To ensure all students had access, we also provided a clone that we had built which served as an interface to ChatGPT but did not require a login.

```
void PrintReverse(char *word)
{
    if (*word == '\0') {
        return;
    } else {
        PrintReverse(word+1);
        printf("%c", *word);
    }
}
```

Figure 1: Function to print a string in reverse, for which students were asked to generate an analogy.

3.3 Data Collection and Analysis

The analogies that students created could connect to any topic or theme of their choosing (see the relevant instructions under 'Themes' in Figure 2). Based on the 841 responses, we observed 456 instances (54.2%) where students did not include the prompt they used to generate the analogy, even though they were instructed to do so. That data was discarded from this analysis.

3.3.1 *Stratifying the Analogies by Prompt Topic.* Using the remaining 385 analogies which contained an analogy and a prompt, we stratified the data based on whether the prompt explicitly specified a topic. This sample sized was used for all analysis, for example, one student used the prompt: *"Can you make an analogy to do with*

Try using ChatGPT to create a few analogies to help explain how the PrintReverse() function uses recursion to print a string in reverse. Your first one might not be that good so try different prompts. Here are some tips:

Clarity. The prompt should clearly define what you want the model to do. Avoid ambiguity. For instance, instead of "Tell me about dogs," use "Provide a detailed description of the characteristics, behavior, and care required for domestic dogs."

Context. ChatGPT responds to the immediate context of the prompt. So, establishing a clear context is crucial. For example, the prompt "Translate the following English text to French: 'Hello, how are you?'" provides clear context and instructions.

Themes. Analogies are most effective when they use themes that you are familiar with or interested in. Consider choosing a theme (like sport, books, or anything else you are interested in) to help generate an interesting analogy.

Precision. Precise prompts yield precise responses. For example, if you want a list, specifically ask for it: "List the top 10 most populous countries in the world."

Choose your favorite analogy and submit it in the space below, along with the prompt (you provided to ChatGPT) used to create it. Please also describe what you like about the analogy you created and why you think it would be helpful to understand the provided code.

Figure 2: Instructions provided to students for Exercise 1.

Pokemon for this code: <code>". The student explicitly specified the topic (**Student-Selected**) to be 'Pokemon'. An example where the student did not include a topic in their prompt is *"provide me with a detailed and concise analogy of the code provided below to explain how the function uses recursion to print a string in reverse"*. In the absence of a student-prescribed topic, ChatGPT will produce an analogy using a topic it selects (**GPT-Selected**), which in this case of this example, was about a labyrinth.

3.3.2 *Identifying the Topics and Themes of the Analogies.* Each of the 385 analogies were coded based on their topic, with the prompts hidden to avoid bias. Each analogy could only be coded with one topic. Next, these topics were grouped thematically into 29 themes, with researchers mediating disagreements as needed. To enhance reliability, the researchers independently recoded all analogies using these themes. A subset of 150 randomly selected rows was used to compute inter-rater reliability using Cohen's Kappa, resulting in 0.83 for GPT-generated topics and 0.84 for student-generated analogies. Subsequent sections refer to these topics and themes.

3.3.3 *Assessing the Diversity of the Analogy Topics.* We measured the diversity of analogy topics generated by ChatGPT by computing the number of distinct topics for the Student-Selected and GPT-Selected analogies. This number of distinct topics was divided by the number of total topics. This calculation resulted in a value between 0 and 1 where a higher number indicates greater diversity of topics across the set.

3.3.4 *Exemplar Analogies.* To exemplify the types of analogies produced by ChatGPT, we chose three exemplar analogies. Our selection process was subjective with the goal to identify analogies that are particularly engaging and insightful. This process involved reviewing a large randomly sampled subset of the analogies. We only

¹https://en.wikipedia.org/wiki/Droste_effect

reviewed analogies with associated student comments to report on students' perspectives about the chosen analogies.

3.3.5 Thematic Analysis of Students' Survey Responses. We analyzed students' open-responses using a reflexive thematic analysis method [5, 6]. This process is 'reflexive' because it acknowledges that the researcher plays an active role in identifying the themes. We followed the six phases which include familiarizing yourself with the data, generating initial codes, searching for themes, reviewing themes, and defining and naming the themes. At each phase, team members discussed the themes to mitigate interpretation biases.

4 RESULTS

4.1 RQ1: Analyzing the Generated Analogies

We found that only 226 prompts (63.1%) contained an explicitly specified analogy topic (*Student-Selected*). The average word count for analogies where the topic was not specified (*GPT-Selected*) was 179.2 words. When students specified the topic, the average word count increased to 235.5 (a 31.4% increase). This suggests that student-specified topics may yield more detailed analogies. In this section, we present examples of analogies generated by students along with their reflections on the analogies. We also compare and contrast analogies in terms of the topic distribution and diversity.

Imagine you're a Pokémon Trainer on an adventure, and you've encountered a special ability called 'Reverse.'

You have a friendly Pikachu by your side. Now, you come across a series of Poké Balls with letters inside them instead of Pokémon. As you walk through the tall grass, you find these letter-filled Poké Balls one by one. You pick up each Poké Ball, and Pikachu helps you examine its content. Instead of revealing the letters immediately, Pikachu tells you to keep it in your backpack for now. You continue to explore, collecting more letter-filled Poké Balls as you go. After a while, you decide to take a break, and Pikachu suggests you start from the last Poké Ball you picked up. You retrieve the last Poké Ball from your backpack and examine it. Inside, you find a letter. You show it to Pikachu, and Pikachu says, 'This letter comes from the last Poké Ball we picked up. Let's keep it.'

You repeat this process, one Poké Ball at a time, starting from the last one you picked up and working your way back through your collection. Pikachu helps you reveal and remember the letters from each Poké Ball.

In the code, your 'PrintReverse' function is like Pikachu's assistance. It processes each character in the string (each Poké Ball), keeps it in memory, and then starts revealing the characters in reverse order, just like you go through your collection of Poké Balls in reverse order, revealing and remembering the letters one by one." ===== PROMPT 2: Can you summarise that into steps?

- STEP 1: You are a Pokémon Trainer on an adventure with your Pikachu, and you come across letter-filled Poké Balls.
- STEP 2: You collect these Poké Balls one by one and store them in your backpack without revealing the letters inside.
- STEP 3: As you continue collecting, Pikachu suggests you start from the last Poké Ball you picked up.
- STEP 4: You retrieve the last Poké Ball from your backpack, reveal a letter, and Pikachu notes that this letter comes from the last Poké Ball collected.
- STEP 5: You repeat this process for each Poké Ball, always starting from the last one and working your way back through your collection.
- STEP 6: Pikachu helps you reveal and remember the letters from each Poké Ball.

In the code, your 'PrintReverse' function follows a similar pattern. It processes each character in the string, keeps it in memory, and then starts revealing the characters in reverse order. Just like in the Pokémon analogy, you work your way backward through your collection of letters, revealing and remembering them one by one.

Figure 3: An Example Analogy Generated by P242

4.1.1 Example Analogies. In this section, we present three personalized analogies generated by ChatGPT in response to prompts that specified a topic. Each analogy aligned with the student's interests and is accompanied by comments from the respective student.

Pokémon Trainer. In this analogy, shown in Figure 3, the student imagines themselves as a Pokémon Trainer with a Pikachu, collecting letter-filled Poké Balls. The analogy provides a step-by-step process for 'collecting' all of the letters and then printing them in reverse order. This scenario is analogous to manipulating a string array where each Poké Ball corresponds to a character variable in a string array. The student (P242) who generated this analogy commented about how LLMs give students the freedom to personalize the analogy topic, which increased their engagement:

*"I like it because its simple, and also fits within the Pokemon world (apart from the talking Pikachu part). Even though its not the most detailed, it gives you a good general idea of what it is. **The freedom of what theme of analogy you can ask... greatly increases the engagement/interest.**" (P242)*

Harry Potter and the Chamber of Secrets. This analogy focused on the journey of Harry Potter through the Chamber of Secrets as an analogy for the traversing character pointers in code. The recursive exploration and revelation in the story are analogous to processing and reversing characters in a string. This student commented on the cultural relevance of this topic for them and other Gen Z students:

*"Using Harry Potter as an example for the analogy for Recursion fits well because **Harry Potter has a cult following and is widely popular amongst Gen Z...** it allows students to understand how recursion works given a familiar context... associating the code with a story like the Chamber of Secrets, the concept of recursion can provide a sense of engagement, which makes the recursion; a difficult concept can be grasped easier than other potential analogies." (P277)*

Russian Nesting Dolls. This last analogy was one of the most popular analogy topics. It uses the analogy of Russian nesting dolls to illustrate recursion. The base case is compared to the smallest doll, with each recursive call opening a larger doll, ultimately leading to the reversal of the order as the process completes. This student talked about how this connected to fond memories as a child and their personal connection to the topic:

*"The reason why I love this analogy so much is it use the concept of unwrapping Russian Doll, which was one of my favourite toys and cartoon when I was young. **In fact that when I heard {Anonymized Instructor} talked about recursion, I immediately remember the Russian Dolls.** The AI successfully utilise that concept and clearly explain how the code use recursion to work which make me understand it without much problem." (P163)*

4.1.2 Comparing the Diversity and Distribution of Topics. As shown in Table 1, when students provided a specific topic for their prompts, the resulting analogies covered a much broader range of topics

Table 1: The top ten analogy themes and diversity of topics for analogies where the topic was specified in the analogy (student-generated) or not specified (GPT-generated).

Student-Generated		GPT-Generated	
Count	Theme	Count	Theme
27	Food/Cooking	53	Russian Nesting Dolls
23	Russian Nesting Dolls	37	Food/Cooking
14	Books	31	Books
13	Sports	22	Cards
9	Miscellaneous	9	Miscellaneous
6	Animals	5	Dominoes
4	Blocks/Legos	4	Mazes
3	Video Games	3	Transportation
3	Dominoes	2	Sports
3	Board games	2	Animals
<i>Distinct Topics: 169</i>		<i>Distinct Topics: 48</i>	
<i>Total Topics: 226</i>		<i>Total Topics: 169</i>	
<i>Diversity of Topics: 0.747</i>		<i>Diversity of Topics: 0.284</i>	

(0.747) than when no prompt topic was provided (0.284). Additionally, there were differences in theme popularity based on the presence or absence of a specified topic in the prompt. This suggests that the default topics that ChatGPT generates when not explicitly directed, do not fully align with the types of topics that students generate. This misalignment in interests suggests that there might be a disconnect between the natural inclinations of ChatGPT and the specific preferences or expectations of students. It raises questions about the effectiveness of default topic generation in meeting user needs, particularly in educational contexts where alignment with students’ preferences aids meaningful engagement. Examples of particularly creative topics specified within students’ prompts included these from P27, P136, P229, P511, and P18:

- “modern-day NSSL phase two launch vehicles”
- “a magic worm that eats the words”
- “Detective Recursive Rex”
- “A grand book that contains the entire history of everything”
- “Conveyor belt of sandwiches”

4.2 RQ2: Analyzing Student Reflections

To understand students’ experiences with the analogy generation activity, we conducted a thematic analysis of post-lab survey responses about the learning activity along with comments about the analogies that students generated. The analysis revealed key themes related to educational value and personal relevance.

4.2.1 Educational Value. When learning complex concepts like recursion, students can often encounter a significant barrier. Despite lectures and coding assignments designed to aid comprehension, many students described a crucial turning point in their understanding frequently emerged through the use of analogies. As P705 describes it, they did not get a ‘real’ understanding of recursion until after the analogies:

“It was initially difficult to wrap my head around recursive functions, and initially, the code I submitted was

actually not recursive. It was mostly after the analogies that I got a real understanding of what recursive functions were. I feel, that the concepts could be explained a bit more in the lectures.”

This feedback is further exemplified by another student’s reflection where they detailed the limited effectiveness of coding tasks alone in grasping complex concepts like recursion. In their experience, simply following procedural instructions was not enough to foster a deep understanding. The student’s revelation occurred while engaging with a multiple analogies:

“I did the coding tasks first, and I didn’t really understand recursion. I was just simply following the preparation document (which was very helpful for this lab). However, doing the analogy section helped me understand recursion a lot better after reading so many analogies lol.” (P584)

The emphasis on ‘so many’ analogies reflects the extensive exploration required to grasp a challenging concept, like recursion, and it also implies that it may be valuable to engage with multiple analogies. These responses highlighted the role that analogies can play in complementing traditional teaching methods. They suggest that the analogies filled students’ knowledge gaps which might not have been fully addressed through conventional teaching methods.

4.2.2 Personal Relevance. Many students described iterating to find the right analogy. For example, 515 tried creating multiple:

“I tried creating few analogies based on different topics (eg. photo and school) before creating this analogy about dogs. However, I think this analogy based on dogs was most interesting compared to other analogies and if this analogy is provided to a person who is interested in pets, this analogy will be a very effective analogy for them to understand how the ‘PrintReverse’ function works. Moreover, this analogy thoroughly explains how recursion works in this function as well.”

Other students shared this opinion. For example, P310 described struggling with an initial analogy because they did not have enough familiarity, but later found an analogy that resonated with them:

“I found this particular response from [ChatGPT] to be the most effective at helping me visualise the recursion process. I initially asked it to provide a response on the topic of sports but its analogy, based around relay running, was fairly confusing. Hence, I asked [ChatGPT] to provide a response based on the theme of history. I found this response to be more engaging as the reference to scrolls helped me further understand how recursion ‘unravels’ itself once the base case is reached.” (P310)

This ability to rapidly create and engage with analogies is a promising benefit of LLMs. Many students described generating multiple analogies and talked about how some made sense to them while others did not.

4.2.3 Overall Student Sentiment. Overall, students were extremely positive about the activity. Based on manually coding the sentiment of 150 randomly selected student reflections about the analogies, we observed only 8 instances (5.3%) contained a negative sentiment, with 7 of those instances also containing positive sentiments. P24 is the only instance that contained purely negative sentiment:

*“This response took quite a number of attempts to produce, because to start **the responses were much too long. This one seems to have enough detail to get an understanding of the topic, but not too much that it becomes boring.**”*

However, with over 90% of the responses containing positive sentiments, it is clear that the analogies were useful for students. In fact, students even described wanting to continue to use ChatGPT to generate analogies in the future. For example, P57 said:

*“Very nice lab on recursion, also the idea of being able to use chatGPT to create analogies is quite a nice thing to think about and **I’ll definitely will be using this analogy part of chat in the future.**”*

5 DISCUSSION

In this study, over 800 students were asked to generate their own analogies as part of an introductory programming lab focusing on recursion. We observed that only 63.1% of the students explicitly specified a topic when prompting ChatGPT. This provided us with the opportunity to stratify the data by the analogies where the topic was explicitly chosen by students and analogies with a topic chosen implicitly by ChatGPT. Without stratifying the data, it would be unclear if students chose the topic. We observed that students tended to come up with more diverse analogy topics on average and the prevalence of topics also varied between the two groups. This disconnect in topical prevalence and diversity suggests that students may benefit from being ‘in-the-loop’ when generating analogies. This aligns with prior work [1, 18]; specifically that analogies should be ‘dialectic, not didactic’ where “analogy use and creation should be guided and discussed, not simply ‘given’” [1]. So while LLMs can easily generate bespoke analogies at any time on-demand, the interests, creativity, and culture of students is necessary to guide the process. In this way, LLMs may provide new opportunities for students to engage with computing concepts in ways that are culturally relevant [12, 31, 32]. Multiple students commented on how the analogies could be relevant for their generation, culture, or lived experiences. This new capability for students to generate personalized analogies with very little effort undermines debates about which analogy is ‘best’ for teaching a specific concept [1]. Theoretically, students could continue to generate analogies until they find one that resonates with them and facilitates their understanding. However, this may add fuel to debates about whether a weak analogy can be misleading or confusing for students. Prior work has shown that even poor analogies do not do “more damage” than not having an analogy [1, 36].

Despite overwhelming positive comments and the consistent advantages highlighted by students, the analysis revealed a surprising trend. Contrary to our initial expectations, a non-negligible number of the analogies provided by students were rather conventional, centering around familiar concepts like books and food.

This raises intriguing questions about the prevalence of certain archetypal analogies in educational settings. One surprising aspect was the limited incorporation of analogies related to individualized interests or niche hobbies, such as a specific musician’s lyrical style or references to movies. Further investigation is warranted to understand this trend better. Potential explanations include students’ reluctance to share personally relevant analogies, the perception that niche topics may be more engaging but less informative, or the possibility that their best analogies, which may not have been the first or the most “personally relevant”, were not initially shared. Additionally, analogies are memorable, which is great, but if they’re inaccurate, this may lead to **memorable misconceptions**. Future research should assess analogy accuracy, focusing on its impact on student trust and the retention of knowledge and misconceptions. These findings raise questions about how much scaffolding students should have in crafting analogies. Should teachers encourage students to generate personally relevant analogies or provide ready-made prompts that allow students to insert a topic.

5.1 Limitations

This study encountered several limitations. Firstly, a large subset of students did not include their prompts along with their analogies, resulting in some data that was not able to be analyzed. While the large sample size helps mitigate these effects, it remains uncertain whether some students faced challenges in generating analogies, and this aspect cannot be examined without the complete dataset. Another limitation is the absence of information on how many analogies students produced before selecting their final one. Multiple students explicitly shared in their reflections that they created analogies for multiple topics. Therefore, the chosen analogy may not fully represent this diversity. Additionally, since students performed this activity as part of a class assignment, the nature of the topics selected might differ if students were generating analogies spontaneously for immediate learning support. Lastly, to streamline the study, we provided students with a single programming problem. It is unclear whether these findings would be replicated with a different problem, as topic representation may vary.

6 CONCLUSION

Our exploratory study investigated the types of analogies that students generated when using large language models (LLMs). Over 350 students generated analogies for a recursive code snippet and provided the corresponding prompt. One consistent finding was that students were largely positive about the learning activity. Students described the value they received regarding the ability to personalize the analogies. Participants described being able to easily remember analogies that were personally and culturally relevant. They also talked about how seeing multiple analogies provided additional value. Despite these benefits, we observed that only half of the students in this study took advantage of being able to personalize the analogies by providing an explicit topic in the prompt. It is possible that some students found sufficient value associated with generic analogies, but more work is needed to understand why so many students failed to specify the analogy topic.

REFERENCES

- [1] Briana Bettin and Linda Ott. 2021. Frozen in the Past: When it Comes to Analogy Fears, It's Time For Us to "Let it Go". In *Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V. 1*. 359–365.
- [2] Briana Bettin, Linda Ott, and Julia Hiebel. 2022. Semaphore or Metaphor? Exploring Concurrent Students' Conceptions of and with Analogy. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 1*. 200–206.
- [3] Briana Bettin, Linda Ott, and Julia Hiebel. 2023. More (Sema| Meta) phors: Additional Perspectives on Analogy Use from Concurrent Programming Students. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*. 166–172.
- [4] Jonas Boustedt, Anna Eckerdal, Robert McCartney, Jan Erik Moström, Mark Ratcliffe, Kate Sanders, and Carol Zander. 2007. Threshold concepts in computer science: do they exist and are they useful? *ACM Sigcse Bulletin* 39, 1 (2007).
- [5] Virginia Braun and Victoria Clarke. 2006. Using thematic analysis in psychology. *Qualitative research in psychology* 3, 2 (2006), 77–101.
- [6] Virginia Braun and Victoria Clarke. 2021. One size fits all? What counts as quality practice in (reflexive) thematic analysis? *Qualitative research in psychology* 18, 3 (2021), 328–352.
- [7] Yingjun Cao, Leo Porter, and Daniel Zingaro. 2016. Examining the value of analogies in introductory computing. In *Proceedings of the 2016 ACM Conference on International computing education research*. 231–239.
- [8] Richard K Coll, Bev France, and Ian Taylor. 2005. The role of models/and analogies in science education: implications from research. *International Journal of Science Education* 27, 2 (2005), 183–198.
- [9] Zijian Ding, Arvind Srinivasan, Stephen MacNeil, and Joel Chan. 2023. Fluid transformers and creative analogies: Exploring large language models' capacity for augmenting cross-domain analogical creativity. In *Proceedings of the 15th Conference on Creativity and Cognition*. 489–505.
- [10] James Finnie-Ansley, Paul Denny, Brett A Becker, Andrew Luxton-Reilly, and James Prather. 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. In *Australasian Computing Education Conf*. 10–19.
- [11] Michal Forišek and Monika Steinová. 2012. Metaphors and analogies for teaching algorithms. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. 15–20.
- [12] Diana Franklin, David Weintrop, Jennifer Palmer, Merijke Coenraad, Melissa Cobian, Kristan Beck, Andrew Rasmussen, Sue Krause, Max White, Marco Anaya, et al. 2020. Scratch Encore: The design and pilot of a culturally-relevant intermediate Scratch curriculum. In *Proceedings of the 51st ACM technical symposium on computer science education*. 794–800.
- [13] Nasser Giacaman. 2012. Teaching by example: using analogies and live coding demonstrations to teach parallel computing concepts to undergraduate students. In *2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & PhD Forum*. IEEE, 1295–1298.
- [14] John K Gilbert and Rosária Justi. 2016. Analogies in modelling-based teaching and learning. *Modelling-based teaching in science education* (2016), 149–169.
- [15] Tina Götschi, Ian Sanders, and Vashti Galpin. 2003. Mental Models of Recursion. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education (SIGCSE '03)*. Association for Computing Machinery, New York, NY, USA, 346–350.
- [16] Colton Harper. 2022. Developing and Evaluating Scaffolding for Student-Generated Analogies in CS1. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol. 2*. 656–657.
- [17] Colton Harper, Ryan Bockmon, and Stephen Cooper. 2023. Investigating Themes of Student-Generated Analogies. In *Proceedings of the ACM Conference on Global Computing Education Vol 1 (CompEd 2023)*. Association for Computing Machinery, 64–70.
- [18] Allan G Harrison and David F Treagust. 2006. Teaching and learning with analogies: Friend or foe? *Metaphor and analogy in science education* (2006).
- [19] Ava Heinonen and Arto Hellas. 2020. Exploring the instructional efficiency of representation and engagement in online learning materials. In *United Kingdom & Ireland Computing Education Research conference*. 38–44.
- [20] Arto Hellas, Juho Leinonen, Sami Sarsa, Charles Koutchme, Lilja Kujanpää, and Juha Sorva. 2023. Exploring the Responses of Large Language Models to Beginner Programmers' Help Requests. In *Proceedings of the 2023 ACM Conference on International Computing Education Research V.1*. ACM.
- [21] Dave Heywood. 2002. The place of analogies in science education. *Cambridge Journal of Education* 32, 2 (2002), 233–247.
- [22] Irene Hou, Sophia Mettillie, Owen Man, Zhuo Li, Cynthia Zastudil, and Stephen MacNeil. 2024. The Effects of Generative AI on Computing Students' Help-Seeking Preferences. In *Proceedings of the 26th Australasian Computing Education Conference (ACE '24)*. Association for Computing Machinery, New York, NY, USA, 39–48. <https://doi-org.libproxy.temple.edu/10.1145/3636243.3636248>
- [23] Anthony Robins Janet Rountree and Nathan Rountree. 2013. Elaborating on threshold concepts. *Computer Science Education* 23, 3 (2013), 265–289.
- [24] Andreas Larsson. 2023. Metaphors and Gestures in Programming Education. In *Programming and Computational Thinking in Technology Education*. Brill.
- [25] Juho Leinonen, Paul Denny, Stephen MacNeil, Sami Sarsa, Seth Bernstein, Joanne Kim, Andrew Tran, and Arto Hellas. 2023. Comparing Code Explanations Created by Students and Large Language Models. *arXiv preprint arXiv:2304.03938* (2023).
- [26] Juho Leinonen, Arto Hellas, Sami Sarsa, Brent Reeves, Paul Denny, James Prather, and Brett A Becker. 2023. Using large language models to enhance programming error messages. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*. 563–569.
- [27] Stephen MacNeil, Paul Denny, Andrew Tran, Juho Leinonen, Seth Bernstein, Arto Hellas, Sami Sarsa, and Joanne Kim. 2023. Decoding Logic Errors: A Comparative Study on Bug Detection by Students and Large Language Models. *arXiv preprint arXiv:2311.16017* (2023).
- [28] Stephen MacNeil, Andrew Tran, Arto Hellas, Joanne Kim, Sami Sarsa, Paul Denny, Seth Bernstein, and Juho Leinonen. 2023. Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book. In *Proc. SIGCSE'23*. ACM, 6 pages.
- [29] Stephen MacNeil, Andrew Tran, Juho Leinonen, Paul Denny, Joanne Kim, Arto Hellas, Seth Bernstein, and Sami Sarsa. 2022. Automatically Generating CS Learning Materials with Large Language Models. *arXiv preprint arXiv:2212.05113* (2022).
- [30] Stephen MacNeil, Andrew Tran, Dan Mogil, Seth Bernstein, Erin Ross, and Ziheng Huang. 2022. Generating Diverse Code Explanations Using the GPT-3 Large Language Model. In *Proc. of the 2022 ACM Conf. on Int. Computing Education Research - Volume 2*. ACM, 37–39.
- [31] Tia C Madkins, Alexis Martin, Jean Ryoou, Kimberly A Scott, Joanna Goode, Allison Scott, and Frieda McAlear. 2019. Culturally relevant computer science pedagogy: From theory to practice. In *2019 research on equity and sustained participation in engineering, computing, and technology (RESPECT)*. IEEE, 1–4.
- [32] Jessica Morales-Chicas, Mauricio Castillo, Ileri Bernal, Paloma Ramos, and Bianca L Guzman. 2019. Computing with relevance and purpose: A review of culturally relevant education in computing. *International Journal of Multicultural Education* 21, 1 (2019), 125–155.
- [33] Laurie Murphy, Sue Fitzgerald, Raymond Lister, and Renée McCauley. 2012. Ability to 'explain in Plain English' Linked to Proficiency in Computer-Based Programming. In *Proc. of the Ninth Annual Int. Conf. on Int. Computing Education Research*. ACM, 111–118.
- [34] James Prather, Paul Denny, Juho Leinonen, Brett A Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, Andrew Luxton-Reilly, et al. 2023. The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*. 108–159.
- [35] Anthony Robins, Janet Rountree, and Nathan Rountree. 2003. Learning and teaching programming: A review and discussion. *Computer science education* 13, 2 (2003), 137–172.
- [36] Yam San Chee. 1993. Applying Gentner's theory of analogy to the teaching of computer programming. *International journal of man-machine studies* 38, 3 (1993), 347–368.
- [37] Kate Sanders and Robert McCartney. 2016. Threshold concepts in computing: past, present, and future. In *Proceedings of the 16th Koli Calling international conference on computing education research*. 91–100.
- [38] Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. 2022. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In *Proc. of the 2022 ACM Conf. on Int. Computing Education Research - Volume 1*. ACM, 27–43.
- [39] Norbert M Seel. 2017. Model-based learning: A synthesis of theory and research. *Educational Technology Research and Development* 65 (2017), 931–966.
- [40] Andrew Tran, Linxuan Li, Egi Rama, Kenneth Angelikas, and Stephen MacNeil. 2023. Using Large Language Models to Automatically Identify Programming Concepts in Code Snippets. In *Proc. of the 2023 ACM Conf. on Int. Computing Education Research - Volume 2*, Vol. 1. ACM, 563–569. <https://doi.org/10.1145/3568812.3603482>
- [41] Camilo Vieira, Alejandra J Magana, Michael L Falk, and R Edwin Garcia. 2017. Writing in-code comments to self-explain in computational science and engineering education. *ACM Transactions on Computing Education* 17, 4 (2017).
- [42] Arto Vihavainen, Craig S Miller, and Amber Settle. 2015. Benefits of self-explanation in introductory programming. In *Proc. of the 46th ACM Technical Symposium on Computer Science Education*. 284–289.
- [43] Jacqueline L. Whalley, Raymond Lister, Errol Thompson, Tony Clear, Phil Robins, P. K. Ajith Kumar, and Christine Prasad. 2006. An Australasian Study of Reading and Comprehension Skills in Novice Programmers, Using the Bloom and SOLO Taxonomies. In *Proc. of the 8th Australasian Conf. on Computing Education - Volume 52*. Australian Computer Society, Inc., AUS, 243–252.
- [44] Cynthia Zastudil, Magdalena Rogalska, Christine Kapp, Jennifer Vaughn, and Stephen MacNeil. 2023. Generative AI in Computing Education: Perspectives of Students and Instructors. In *2023 IEEE Frontiers in Education Conference (FIE)*. 1–9. <https://doi.org/10.1109/FIE58773.2023.10343467>