EXPLORING AI-DRIVEN PROGRAMMING EXERCISE GENERATION

Juuso Rytilahti, Oshani Weerakoon, Erkki Kaila

Department of Computing, University of Turku

ABSTRACT

Large language models(LLMs) are transforming how teachers work. In this paper, we observe several experimental approaches to generating software programming exercises by utilizing ChatGPT, a popular and open LLM. The generation of these exercises was tightly connected to a large Python programming course that was targeted at students studying in Information Technology, Software Engineering, and Computing.

We experimented with three separate approaches. In the first one, we generated new programming exercises with a specific topic using theme injection. In the second one, we generated variations of existing programming exercises by changing the theme or content. In the third one, we generated hybrid exercises by injecting original programming exercises with additional topics or other related exercises.

Based on our results, all three approaches showed potential but also revealed limitations. The exercise generation with theme injection can produce fully functional exercises. However, these exercises could appear to students as too generic or erroneous. The exercise variations seem to retain the semantic meaning of the original exercise quite well while still using different context. We also tested the variations in a large introductory programming course and found out that the students could not distinguish them from human-generated exercises in style or quality. The hybrid exercises were built upon the idea of exploring how close we are to fully adaptive learning environments in the field of programming education. The current results of this approach show that we need to do further experimentation to maybe reach the goal.

All in all, it was evident that LLMs can be a useful tool in assisting teachers in generating exercises. Even with certain coherent limitations, they are useful in particular cases. We conclude our article by discussing the future possibilities of LLMs, including but not limited to dynamic, automatically generated exercises and fully adaptive learning environments.

KEYWORDS

AI in Education, LLM, Programming Education, Pedagogical Tools, ChatGPT, Standards: 2, 3, 5, 8, 9, 11

INTRODUCTION

The field of generative AI has advanced rapidly in the recent years. These innovative AI tools are revolutionizing work processes across a wide range of industries by automating routine tasks. The field of teaching and education represents a prime example of an area where the potential of generative artificial intelligence has been a topic of conversation. In the study conducted by Nelson and Creagh (2023), it has been observed that the rapid development of generative AI tools has sparked numerous discussions and advice forums on how best to integrate them into both teaching methods and assessment practices.

ChatGPT is a Large Language Model(LLM), trained to answer a prompt given by the user and it is based on a similar architecture as InstructGPT (Ouyang et al., 2022). These models can perform well in different tasks, e.g. code generation and text summarization (Bubeck et al., 2023). These systems can now generate coding exercises and even multimodal models are being trained, E.g. Gemini by Google DeepMind. Team et al. (2023).

Teachers are often pressured for time. Teacher's workload can be further increased by large student groups, or tight schedules not to mention creating and updating study material for courses. ChatGPT and other new LLMs can help to reduce the time needed for content generation for courses. To reduce workload of the teachers and to explore the capabilities of these new models, we have explored the possibilities of how to utilize AI when creating programming exercises. The performance of these models also sparks a question of which parts of the programming education can be automated with these new AI tools.

The paper is structured as below. Section II describes related studies conducted by different researchers across the world and Section III explains the methodology that we used to conduct these experiments. Section IV describes a set of limitations that we observed during the exercise generation. Section V discusses the results and observations in detail and the Final Section concludes the findings of our study.

RELATED WORK

Crawford, Cowling, and Allen (2023) suggests that educators can utilize AI tools such as ChatGPT to create supportive learning environments. The paper acknowledges existing literature on plagiarism and academic integrity and discusses the role of leadership in supporting the ethical use of AI. In Sovietov (2022), the authors focus on the automatic generation and grading of programming exercises. It describes the general scheme for constructing a programming exercises generator, highlighting two classes of exercises that can be automated: converting notation into code and converting data formats.

Another study conducted by Wang, Singh, and Su (n.d.) discusses the "Search, Align, and Repair" (SARFGEN) data-driven program repair framework for automating feedback generation in introductory programming exercises. The framework aims to provide efficient, fully automated, and problem-agnostic feedback for large-scale or MOOC-style courses by leveraging a large number of student submissions.

Speth, Meissner, and Becker (2023) has investigated the use of AI models, like ChatGPT, for creating exercises for programming courses. It involves creating exercise sheets with ChatGPT for a beginner to intermediate programming course and assessing the quality of these exercises in an actual course setting.

Despite all the capabilities of AI tools like ChatGPT, it should also be highlighted the possible risks that generative AI has introduced to the field of education. The study by Nelson and Creagh (2023) discusses valid concerns regarding the integrity of assessments and the potential reputational risks for educational institutions with the introduction of AI tools. It further encourages educators and researchers to explore how these tools can be used effectively and ethically to enhance learning outcomes and student capabilities.

STUDY SETTING AND METHODS

Our approach was explorative. During our experimentation with AI-driven programming exercise generation, we decided to focus on three distinct approaches.

1. Generating exercises from a specific topic with theme injection,
2. Generating variations from existing exercises by changing the context of the original exercise, and
3. Generating hybrid exercises by injecting original exercise with additional topics or with other exercises.

All of these approaches have different use cases, strengths, and limitations. we have experimented with both models of ChatGPT (GPT-3.5 and GPT-4). As a more general note, we found that GPT-4 tended to create better, more "creative" exercises. The examples presented in this study are generated with GPT-4. We also note that the performance seemed to have improved since the original experimentation was done in the summer of 2023.

Generating exercises from a specific topic with theme injection

The first approach was to generate whole new exercises from a given topic and theme. The main drawback of this method seems to be that the exercises are often shallow and generic or lack the necessary supplementary materials. Still, the approach can be useful. An example of a prompt that is used can be seen in Table 1.

It seems to be possible to use almost any theme successfully, as we experimented for example with *the medieval times* or *working in an office* as themes. Due to the nature of LLMs, the quality of generated exercises can vary heavily. However, sometimes just re-generating the exercise with the same prompt provided sufficiently better results. Providing more context seemed to always improve the quality of generated exercises. One example of utilizing more descriptive context can be seen in Table 2.

Table 1.shows an example of generating exercise from a specific topic with theme injection. The most relevant part of the prompt is the last paragraph. In this particular example, the topics have also been generated by ChatGPT, the topic generation prompt can be seen here. The full chat can be viewed here.

Table 1. Example of generating exercise from a specific topic with theme injection.

Type	Description
Input	File Handling: Reading from and writing to files is a useful skill for many practical applications. Exercises List: • Exercise 1: Write a program that reads a text file and counts the number of words. • Exercise 2: Create a log file writer that records timestamps and custom messages. Create exercise descriptions for the above exercises. Let the exercise descriptions revolve around the theme of working in a cafe.

Table 2.shows an example of generating a Javascript exercise for students in a given university course. The objective of the exercise is to providing students with an inefficient code which is then required to revamp to a more efficient code by following green programming practices. The full chat can be viewed here.

Table 2. Example of generating a Javascript exercise for students in a given university course.

Type	Description
Input	Assume you are a university teacher who is teaching Green Programming to students. You are required to come up with an exercise to utilize good green programming practices in coding. Let's create a bad code example in javascript where the code has not followed the 5 of the best practices mentioned in the given chapter text. The final code should not be complicated, that even a student with low coding competency could understand. Chapter content is given below for your referral. <chapter content>

Generating variations of existing exercises

Generating variations from existing programming exercises seems to be a very promising approach in exercise generation, as in our experiments it has produced usable results consistently. This generation can be used in different settings, for example for generating exercise variations for exams, or to display a more abstract problem set. These exercises could also be used to quickly generate additional exercises, even for an ongoing course.

This is the prompt we used to generate variations:

- "*You are a university teacher teaching computer science. Below is an exercise. Create variations of the exercise description below. These variations will be used in the exam on introduction to programming course. The variations need to convey the same or almost same meaning but use different context. Answer in Finnish.*"

We noticed that this approach tends to perform best when there is a lot of context tied to a certain theme that can be changed. If the exercise contained mechanical instructions only, the variation often just paraphrased the original exercise and did not change (or introduce) a

theme. In this kind of case, injecting a theme into the prompt often produced a better quality variant.

Can students identify the LLM-generated variants?

To test the quality of variants, we swapped 3 exercises to AI-generated variants in a final week module in an introductory programming course. A small survey was included where the students were asked if they identified the AI-generated exercises. The generated exercises were in Finnish, and only a couple of very minor grammatical fixes were made manually to match the wording of other exercises (i.e. "Write a function" instead of "Implement a function"). A total of 378 students answered the survey. The distribution of the submitted answers can be seen in Figure 1.

Additionally, the students were asked to describe how they detected the AI-generated exercises. The common reasons listed by the students were differences in the exercise formatting, spelling, structure, or tone of the exercise. The common nominator in the correctly identified AI-generated exercises was the differences in the wording and style formatting of the exercise. Interestingly, spelling mistakes and different and missing style formatting were also emphasized as reasons identifying the exercise as AI-generated in the wrong answers submitted by students. As seen in the figure, the vast majority of students were unable to identify AI-generated exercises from original ones. This result could probably be increased further by making sure that the AI-generated exercises have similar formatting, structure, and wording as other exercises in the course.

Generating hybrid exercises

Hybrid exercises are the best showcase for the limits of the current generation of LLMs. The definition of a hybrid exercise is that you would have two exercises, or one exercise and an additional topic that could be uniquely combined to create a new exercise. Ideally, this new exercise would use a different theme, and solving it would require the skills that the exercises used to create it utilized.

Our original prompt produced mostly poorly constructed results, as the exercise descriptions it generated were often a bit nonsensical or illogical. Quite often, instead of creating a new, fully unique exercise, the combined exercise was simply a listing of both exercise descriptions one after another with rather slight modifications. The original prompt can be seen below:

- "*You are university teacher teaching computer science and software engineering. You are creating exam for students of introduction to programming course. Below are two exercise description. Combine them in an unique way in a new context.*
  Exercise 1:
  <exercise description>
  Exercise 2:
  <exercise description>"

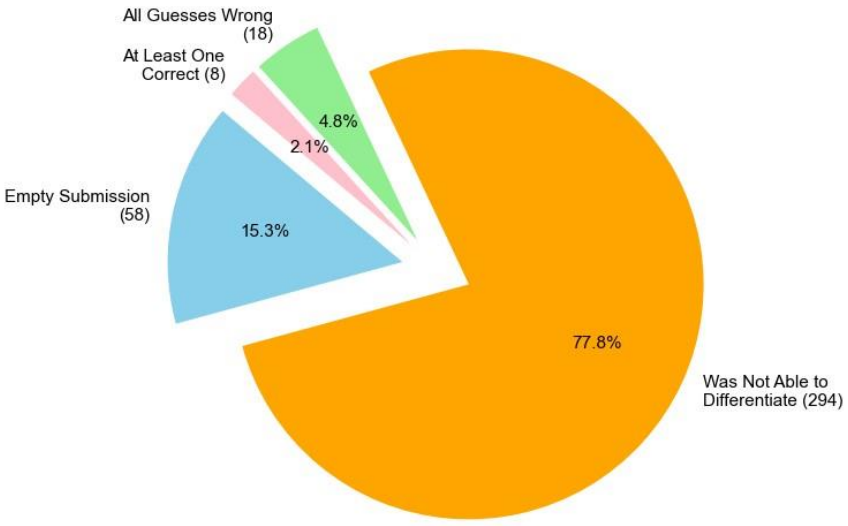

Figure 1. Answers to the survey about detecting AI-generated exercises (N=378).

No student could identify all three variations correctly. The actual amount is displayed inside parentheses. Half of those who identified at least one correctly tagged also exercises written by a human as AI-generated.

However, after we modified the prompt by utilizing a more well-defined and rigid structure, and made it utilize a chain-of-thoughts, the quality of the generated exercises seemed to increase significantly. A prompt used to generate a hybrid exercise can be seen below:

- "*You are a university teacher teaching computer science and software engineering. You are creating exams for students of introduction to programming course. Below are two exercise descriptions. Uniquely combine them in a new context. First choose a new theme for the exercise, then explain your rationale on how you will combine the exercises, and then finally offer the final answer of the combined exercise.*
  Exercise 1:
  <exercise description>
  Exercise 2:
  <exercise description>"

Still, we would like to note that our experimentations with hybrid exercise generation have still been fairly limited quantitatively and more research is needed for the topic.

Limitations

The primary language used in ChatGPT's training data is English, which may affect its effectiveness in other languages Dave (2023). ChatGPT tends to generate convincing but inaccurate information, a phenomenon known as 'hallucination' Bubeck et al. (2023), and can also reinforce stereotypes. There are notable variations among different language models; for instance, GPT-4 generally surpasses GPT-3.5 across a broad spectrum of tasks OpenAI (2023). The proficiency of ChatGPT in executing tasks is also influenced by the subject matter and the clarity of the input prompt.

This is preliminary work, so more research is needed to find out the limitations of these approaches. Furthermore, quantifying the results we found with a more statistical approach could provide valuable insight into the limitations of these methods. We would like to also add, that utilizing more advanced prompting techniques could potentially improve the results further.

DISCUSSION

The field of automatic programming exercise generation shows rapid advancements. However, the limits of methods discussed in this study are still a bit unclear. One should also remember that utilizing ChatGPT (or other LLMs) might have surprising caveats. For example, it might favor the first presented option when asked to compare multiple options (Dettmers, Pagnoni, Holtzman, and Zettlemoyer (2023)). Deng and Lin (2022) also point out in their study that ChatGPT may repeat biases or offensive language due to the data used in the model's training. Even with these potential fallbacks, the ease, speed and, surprisingly good quality of generated exercises show that LLMs can be valuable assets for a teacher.

In variation generation from existing exercises, we have not yet explored how well the models could fit the existing unit tests for exercises. They might perform quite well since the semantic information regarding new tests should be quite similar to the old exercise's tests. Another question is whether the models can do more complicated test modifications, e.g. when a datatype is changed from string to a number (for example when the exercise theme changes from the music to movie ratings) in the generated variation.

Even though in hybrid exercise generation the achieved results leave much to be desired, the foundation is still there. Defining a guideline for producing good programming exercises can be a difficult task since the guidelines will probably at least partly lean into values that can be seen as abstract, such as *creativity* or *innovativity* or *enjoyability*. Thus, we can argue that understanding the quality of a programming exercise can be a challenging task for a machine, at least for a while. It should also be noted that taking two arbitrary exercises and combining them can be a challenging and time-consuming task even for the most experienced teachers.

It should also be noted that ChatGPT is not deterministic. The answer given by ChatGPT in different threads can be the same, or paraphrased or it might even present an opposite result of a previously given answer in a different thread. When these approaches are directly used by the teacher, not by a student, the presented approaches do not need to work *every* time, but instead *most* of the time. Teacher can keep on generating new exercises until a suitable one comes along. Naturally, in an adaptive system where the exercises need to be generated on the fly, this is not possible.

CONCLUSION

In this paper, we have explored the AI-driven programming exercise generation from multiple possible approaches. Although there are limitations, the chosen approaches show a lot of potential. Many of the generated exercises and exercise variants seem to be usable, especially after minor human-made modifications. It should be noted, that when the AI-generated exercise variants were used at the end of a large-scale programming course, the students who had already completed more than 150 human-made exercises, mostly could not distinguish between AI-generated variants and original exercises.

The different approaches presented in this study performed well. The theme injection's biggest challenge was to generate supplementary materials for the generated exercises. The exercise variation generation showed consistent results with high-quality output. The performance in hybrid exercise generation varied wildly depending on different factors, e.g. on the exercises used as input, but even the initial approach presented in this study could provide (sometimes) good results.

While we can say that fully adaptive learning systems are not here yet, approaches presented in this study can already be useful to teachers in a variety of different settings. Additionally, the applicability beyond the realm of computer science might be possible at least in some application areas, e.g. in generating math word problems. Furthermore, combining the techniques introduced in this study might create interesting results. More research, especially involving more advanced prompting techniques such as tree-of-thoughts is needed, and it is something we will probably address in a future paper.

As AI starts to play a bigger role in teaching methods, it's important for educators to stay alert. They need to make sure that the content AI creates is accurate and appropriate, and this responsibility falls on the person who creates the content to double-check for any mistakes or biases. The AI methods that we have experimented within this study look promising, but we need more research to understand their limits and the most prominent use cases. However, as the technology is rapidly developing, this may be challenging.

FINANCIAL SUPPORT ACKNOWLEDGEMENTS

The authors received no financial support for this work.

REFERENCES

Bubeck, S., Chandrasekaran, V., Eldan, R., Gehrke, J., Horvitz, E., Kamar, E., . . . Zhang, Y. (2023). Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Crawford, J., Cowling, M., & Allen, K.-A. (2023). Leadership is needed for ethical chatgpt: Character, assessment, and learning using artificial intelligence (ai). *Journal of University Teaching and Learning Practice*. doi: 10.53761/1.20.3.02

Dave, P. (2023, May 31). *Chatgpt is cutting non-english languages out of the ai revolution.* https://www.wired.com/story/chatgpt-non-english-languages-ai -revolution/. WIRED.

Deng, J., & Lin, Y. (2022). The benefits and challenges of chatgpt: An overview. *Frontiers in Computing and Intelligent Systems*, *2*(2), 81–83.

Dettmers, T., Pagnoni, A., Holtzman, A., & Zettlemoyer, L. (2023). Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*.

Nelson, K., & Creagh, T. (2023). Editorial volume 14 issue 1 2023. *Student Success*. doi:

10.5204/ssj.2860

OpenAI. (2023). *Gpt-4 technical report.*

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., . . . Lowe, R. (2022). Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, *35*, 27730–27744.

Sovietov, P. (2022). *Automatic generation of programming exercises.* arXiv.

Speth, S., Meissner, N., & Becker, S. (2023). Investigating the use of ai-generated exercises for beginner and intermediate programming courses: A chatgpt case study. In *Csee&t 2023.* (This study

explores the use of AI, particularly ChatGPT, in creating programming exercises and assesses their effectiveness in an educational setting.)

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., . . . Ahn, J. (2023). Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Wang, K., Singh, R., & Su, Z. (n.d.). *Search, align, and repair: Data-driven feedback generation for introductory programming exercises.* https://arxiv.org/. Retrieved from https://arxiv .org/pdf/1711.07148.pdf

BIOGRAPHICAL INFORMATION

Juuso Bernhard Rytilahti is a Research Assistant at the Department of Computing at the University of Turku. His interests are adaptive learning environments and the utilization of generative AI, especially large language models, in education and programming.

Oshani Sisara Weerakoon works as a Research Assistant at the Department of Computing at the University of Turku. She is currently pursuing her master's in Software Engineering at the University of Turku. Her research interests lie primarily in the area of software engineering and interaction designing, more specifically in the usage of generative AI for education.

Erkki Kaila is a university lecturer at the University of Turku. He did his PhD about utilizing technology in programming education. His research interests include learning analytics, pedagogy in programming, ethics of technology usage, and AI in education.

Corresponding author

Juuso Bernhard Rytilahti
Dept. of Computing
20014 Turun yliopisto
Finland jubery@utu.fi