Authoring Programming Exercises for Automated Assessment Assisted by Generative AI

Yannik Bauer
DCC – FCUP, Porto, Portugal

José Paulo Leal
CRACS – INESC TEC, Porto, Portugal
DCC – FCUP, Porto, Portugal

Ricardo Queirós
CRACS – INESC TEC, Porto, Portugal
uniMAD – ESMAD, Polytechnic of Porto, Portugal

Abstract

Generative AI presents both challenges and opportunities for educators. This paper explores its potential for automating the creation of programming exercises designed for automated assessment. Traditionally, creating these exercises is a time-intensive and error-prone task that involves developing exercise statements, solutions, and test cases. This ongoing research analyzes the capabilities of the OpenAI GPT API to automatically create these components. An experiment using the OpenAI GPT API to automatically create 120 programming exercises produced interesting results, such as the difficulties encountered in generating valid JSON formats and creating matching test cases for solution code. Learning from this experiment, an enhanced feature was developed to assist teachers in creating programming exercises and was integrated into Agni, a virtual learning environment (VLE). Despite the challenges in generating entirely correct programming exercises, this approach shows potential for reducing the time required to create exercises, thus significantly aiding teachers. The evaluation of this approach, comparing the efficiency and usefulness of using the OpenAI GPT API or authoring the exercises oneself, is in progress.

2012 ACM Subject Classification Applied computing → Computer-assisted instruction; Computing methodologies → Artificial intelligence; Applied computing → Interactive learning environments

Keywords and phrases ChatGPT, generative AI, programming exercises, automated assessment

Digital Object Identifier 10.4230/OASIcs.ICPEC.2024.21

Funding This work is co-funded by the Erasmus+ Programme of the European Union within the project FGPEPlusPlus, with Agreement Number 2023-1-PL01-KA220-HED-000164696.

1 Introduction

Since its introduction in late November 2022, ChatGPT has produced a range of reactions, from enthusiasm to warnings. In academic and educational contexts, there is legitimate concern about the potential impacts of extensive language models and generative AI. It is important to acknowledge and address these concerns, but it is equally important to recognize the potential of these tools to generate text, code, and data, offering valuable resources and innovative approaches that can positively enhance the educational landscape.

This paper explores the utility of ChatGPT in facilitating the creation of programming exercises, particularly those designed for automated assessment. The process of authoring these exercises is time-consuming and error-prone. It involves generating three distinct components: exercise statements articulated in a natural language, such as English; solutions in a programming language, such as JavaScript; and test data, including input and expected output files. Generative AI is promising for faster and more accurate exercise creation.

An initial experiment was conducted within the Framework for Gamified Programming Education (FGPE+) project [13] aiming to create 120 programming exercises suited for automated evaluation for AuthorKit, a programming exercise repository [15]. This experiment highlighted ChatGPT's difficulties in generating valid JSON files and creating effective and functional test cases for the proposed code solutions. Learning from this, an enhanced feature was developed for Agni, a virtual learning environment, to assist teachers in creating programming exercises [20]. An ongoing evaluation is comparing the time, quality, and benefits of creating exercises independently versus using ChatGPT's assistance.

2 Related Work

The integration of generative artificial intelligence (AI) in education has gained increasing attention due to its potential to transform teaching and learning practices. Indeed, for teachers, writing good questions/exercises and test cases is a fundamental and time-consuming challenge [11, 24]. New AI tools have already increased productivity in work environments, as measured in a study of issues resolved per hour. They found an increase of 14% on average, including a 34% improvement for novice and low-skilled workers, but with minimal impact on experienced and highly skilled workers [4]. A variety of studies have discussed the possibilities of AI in generating educational content and providing personalized learning experiences, highlighting both the opportunities and challenges of these technologies [3, 8, 2, 1, 14, 18, 23].

A study by Sarsa et al. analyzed the quality of programming exercises generated by OpenAI Codex. It revealed that 84.6% of the exercises generated included a sample solution, with 89.7% of these being executable. Furthermore, 70.8% of these exercises featured tests, although only 30.9% of them passed all tests, achieving a test coverage rate of 98.0% [22]. Similarly, another study showed that 75% of exercises created by generative AI were sensible, 81.8% novel, and between 75.8% to 79.2% aligned with the intended themes and concepts [6]. Another study evaluated that students perceived exercises generated by AI as equal to those created by people. However, the limited variety in AI-generated examples and their close adherence to given prompts raise questions about their adaptability in creating diverse learning resources [5]. These findings suggest that while generative AI can produce good primary educational content, its quality and reliability are still lacking [19].

Other uses, such as automatic feedback, have also shown promise. A web application that leverages GPT-4 to provide feedback on complex exercises demonstrated a high correlation with human feedback and deviated by only 6% from human evaluations [10]. A study on the customization of learning content via generative AI found that AI-generated materials positively impacted lower-performing students without negative effects on accessibility [17]. The findings highlighted the benefit of personalized learning experiences, particularly for students struggling with subjects, by providing materials suited to their specific needs, while more proficient students received more advanced content.

In conclusion, while generative AI holds significant potential for education, it must be integrated carefully, considering its abilities and limitations. Empirical studies and responsible guidelines are essential for harnessing the benefits of AI in educational settings.

3 Early exploration

Within the FGPE+ project, an experiment was conducted using OpenAI's GPT-3.5 API to automatically generate 120 programming exercises. These exercises were designed for AuthorKit, a platform for developing programming exercises, that support the definition

Table 1 Results of using GPT to generate exercises.

	Total	% of Total	% of Valid	Resolved	% of Resolved
JSON errors	199	35%		23	12%
Code errors	245	44%	67%	3	1%
Correct exercises	120	21%	33%		
Total	562	100%			

of various parameters. The experiment aimed to generate exercises with several parameters including title, difficulty level, context, task description, input and output details, an example, the language used for the solution, the solution code itself, and five input/output tests. Moreover, parameters such as title, context, etc., were required in multiple languages: Portuguese, English, Italian, and Polish. The objective was to use the GPT API to automatically generate these exercises, verify their correctness, and convert them into the YAPExIL format [16], making them suitable for further integration into AuthorKit.

The main challenges in this process involved getting consistent responses for automatic conversion and testing, as well as ensuring the correctness of the programming exercises. Several different approaches were initially attempted, such as simply asking for the exercise with the desired information or requesting separation with ";", among others. All these attempts faced the issue of GPT providing inconsistent responses, with changing field keys and sometimes breaking the requested format. Finally, it was decided to request the responses in JSON format, providing an example within the prompt. Thus, each prompt included a description of the required parameters and an example of a response in JSON format. Once the exercises were received, the JSON format was validated. If the format was valid, the solution code was tested with the input and output tests. Therefore in the process, two types of errors could occur: a JSON error, as illustrated in Listing 1, indicating a problem with the format, or a code error, as shown in Listing 2, arising from issues in the solution code or failure of a test case. When one of these errors occurred, a response was sent to GPT describing the error and requesting a fixed exercise. This process is showcased in Figure 1.

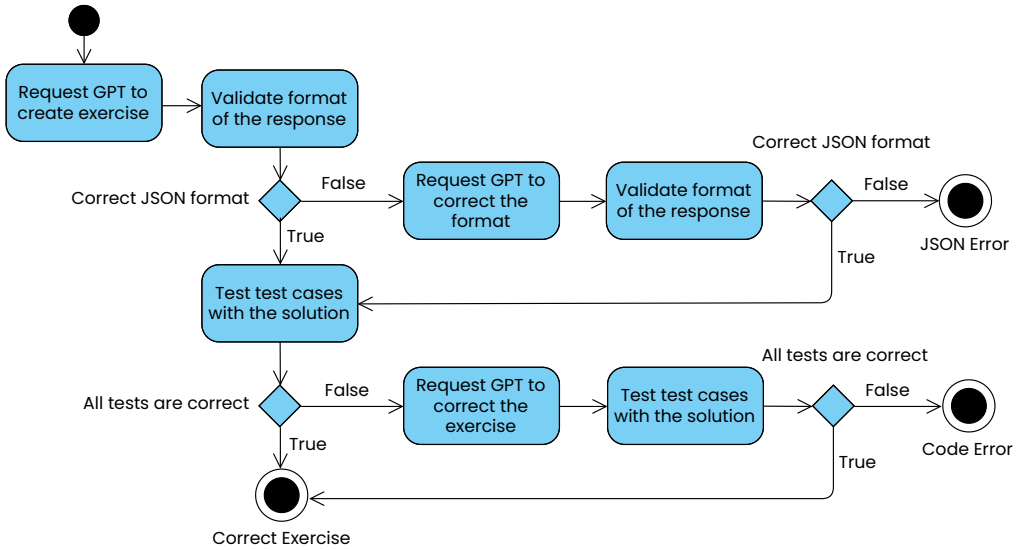

Figure 1 Process of the experiment.

Table 1 displays the results of achieving 120 correct programming exercises, including the number of JSON and code errors, as well as those that were resolved. To arrive at 120 accurate exercises, a total of 562 were generated, indicating that only 21% of the exercises were correct. 199 exercises, accounting for 35% of the total generated, encountered JSON Errors, of which 12% were rectified upon consulting GPT. Furthermore, 245 exercises, which represent 44% of the total, failed due to errors in a test case or the solution code. Only 3 of these could be corrected. It is important to note that a code error can only be detected if the JSON format is valid. This means that out of the 363 exercises with a valid JSON format, 245 had a code error, translating to 67%. The consistency of parameters such as title, solution, task, etc., was manually verified across most generated exercises, and no significant discrepancies were found.

Some limitations of the experiment included the use of GPT version 3.5; it is possible that version 4.0 might have delivered better results. The reason behind choosing GPT 3.5 was that when the experiment was conducted, GPT 4.0 was relatively new and associated with a higher cost. Furthermore, at that time, there were no other well-known generative AI options with an API to choose from. Additionally, generating the parameters step-by-step could have potentially enhanced GPT's capabilities, but this approach was not chosen due to the API's limitation of three requests per minute [7, 21]. Not assigning a specific role to GPT in the prompts, which is often suggested to improve outcomes, was another constraint. Furthermore, the generation of numerous parameters and the extensive size of the prompts might have negatively impacted the results.

Listing 1 Example of generated exercise with a JSON Error (missing brackets at the end).

```
{
"title":{"english":"FizzBuzz problem with a twist", ...},
...
"task":{"english":"Write a function that given n, prints the
    FizzBuzzBang output from 1 to n", ...},
...
"solution_code":"...",
"tests":[..., {"input":"1", "output":"1"}
```

Despite these limitations, the results highlight a challenge with GPT in generating valid JSON formats using the chosen method. Similar issues with JSON formats have been reported by other programmers in community forums [9]. The findings also underscore GPT's limitations in creating completely accurate programming exercises. With a general code error rate of 44%, and 67% when considering only exercises with valid formats, the experiment clearly demonstrates these constraints. However, it is worth noting that in many code error instances, the overall exercise was correct; often, only one or two test cases were slightly off, sometimes due to interpretation issues. For instance, in tasks asking for the maximum word length in a string, there were discrepancies in whether a comma was counted to the word before or not. In the solution code, GPT generated a code that counted a comma with the word before, however, in the test case, it interpreted it contrarily.

4 Authoring Tools

GPT was integrated into Agni [20], a web-based platform designed for learning and teaching JavaScript, to assist teachers in creating programming exercises. However, some changes were made from the previous experience to improve the results. The approach of Joseph

Listing 2 Example of generated exercise with a Code Error (first test case).

```
{
"task":{"english":"Create a function that receives a sentence and
    returns the largest word in it. If there are multiple words with
    the same length, return the first one in the sentence.", ...},
...
"solution_code":
    "def largest_word(sentence):
        words = sentence.split()
        largest = ''
        for word in words:
            if len(word) > len(largest):
                largest = word\n return largest",
"tests":[
    {"input":"Hello, world!", "output": "world"},
    {"input":"What is your profession?", "output": "profession?"},
    {"input":"Mathematics isn't a sport.", "output": "Mathematics"},
    ...]
}
```

Martinez [12] was followed to include a JSON schema as a parameter in the API prompt. This clearly improved the consistency of the responses with no further JSON errors detected. Also, GPT was instructed to take the role of a teacher to help create programming exercises for JavaScript. Another distinction from the experiment was the demand for viewer parameters in Agni, accommodating requests for exercise titles, statements, solution codes, and corner test cases with input and expected output. The version of GPT, namely GPT-3.5, was kept.

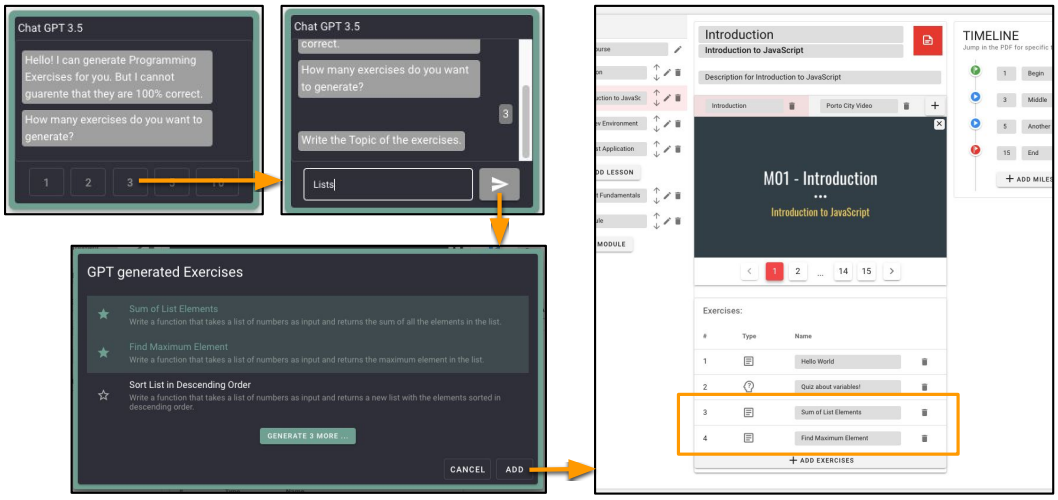

Figure 2 Interface of Agni with the Chat GPT Chatbot.

To provide users with an interactive experience and considering broader future use cases with AI, a chatbot-like feature was incorporated, as shown in its usage in Figure 2. In the early stages, this feature does not provide a direct conversation with ChatGPT but serves as a means to input data to generate exercises. This "bot", accessible during the editing process of a lesson, notifies authors of potential inaccuracies in the exercises, which can occur despite improvements made from previous experiments. It then prompts users to specify the number and topic of exercises they wish to create. A request is sent to the OpenAI API to create the

specified number of programming exercises on the desired topic. The received exercises are then automatically validated in JSON format using the Ajv JSON Schema validator. If valid, they are presented in a pop-up displaying key information such as the title and statement. Within this pop-up, authors can request the generation of three additional exercises and select those to be included in the current lesson. Upon selecting the exercises and clicking the button to add them, they are added to the current lesson. The evaluation of the test cases and the solution is not performed in this feature because, in the previous experiment, many code errors occurred when only one or two out of five test cases failed due to minor issues. Therefore, it was deemed more effective to keep all exercises, as those with errors can be easily corrected. After adding the exercises, when a teacher enters one of the generated ones, the solution code with its test cases is run showcasing visually to the teacher which of the tests fail or pass.

This refined approach resolved the formatting challenges of the initial experiment, offering a robust tool to aid teachers in authoring programming exercises.

5 Ongoing and Future Work

This paper presented an experiment on creating programming exercises using the OpenAI API, highlighting its difficulties in generating valid JSON formats and producing sample solutions with correct test cases. In the experiment, 67% of the correctly formatted exercises contained a code error, indicating that a test case failed when running the provided solution. However, it is important to note that often it was only one out of five test cases that failed, sometimes due to an interpretation issue of the exercise. A refined feature to assist teachers in generating programming exercises was then integrated into Agni. To address the formatting issues, a different approach was adopted, which involved providing a JSON schema in the prompt to the OpenAI API. However, challenges with the solutions and test cases persisted, which is why the feature is considered an assistant rather than an automatic generator.

Despite the stated challenges, the feature still offers potential gains in the speed of creating new programming exercises. Consequently, a questionnaire and survey are being conducted where evaluators have to create programming exercises using Agni, both with and without the help of GPT. The evaluators are recording their time and then responding to follow-up questions about their opinions on the quality and efficiency of using GPT compared to not using it.

After collecting a substantial number of responses, the data will be analyzed to understand the teachers' opinions on this tool and whether a significant reduction in time was observed. We expect these findings to provide insights into how teachers perceive generative AI features and to help to improve current and potentially future features.

References

1 Eman A. Alasadi and Carlos R. Baiz. Generative ai in education and research: Opportunities, concerns, and solutions. *Journal of Chemical Education*, 100(8):2965–2971, 2023. doi:10.1021/acs.jchemed.3c00323.

2 Brett A Becker, Michelle Craig, Paul Denny, Hieke Keuning, Natalie Kiesler, Juho Leinonen, Andrew Luxton-Reilly, James Prather, and Keith Quille. Generative ai in introductory programming. *Name of Journal*, 2023.

3 Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. Programming is hard - or at least it used to be: Educational opportunities and challenges of ai code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2023, pages 500–506, New York, NY, USA, 2023. Association for Computing Machinery. doi:10.1145/3545945.3569759.

4 Erik Brynjolfsson, Danielle Li, and Lindsey Raymond. Generative ai at work, 2023. arXiv:2304.11771.

5 Paul Denny, Hassan Khosravi, Arto Hellas, Juho Leinonen, and Sami Sarsa. Can we trust ai-generated educational content? comparative analysis of human and ai-generated learning resources, 2023. arXiv:2306.10509.

6 Paul Denny, Sami Sarsa, Arto Hellas, and Juho Leinonen. Robosourcing educational resources – leveraging large language models for learnersourcing, 2022. arXiv:2211.04715.

7 Enterprise DNA Experts. How to use chat gpt: A simple guide for beginners, September 2023. URL: https://blog.enterprisedna.co/how-to-use-chat-gpt/.

8 Stefan Feuerriegel, Jochen Hartmann, Christian Janiesch, and Patrick Zschech. Generative ai. *SSRN Electronic Journal*, January 2023. doi:10.2139/ssrn.4443189.

9 Aaron Issac. Chatgpt functions malformed json, July 2023. URL: https://community.openai.com/t/chatgpt-functions-malformed-json/306509.

10 Lukas Jürgensmeier and Bernd Skiera. Generative ai for scalable feedback to multimodal exercises in marketing analytics. *Available at SSRN*, 2024.

11 Richard Lobb and Jenny Harlow. Coderunner: a tool for assessing computer programming skills. *ACM Inroads*, 7(1):47–51, February 2016. doi:10.1145/2810041.

12 Joseph Martinez. Return json from gpt, July 2023. URL: https://betterprogramming.pub/return-json-from-gpt-65d40bfc2ef6.

13 Rytis Maskeliūnas, Robertas Damaševičius, Tomas Blažauskas, Jakub Swacha, Ricardo Queirós, and José Carlos Paiva. Fgpe+: The mobile fgpe environment and the pareto-optimized gamified programming exercise selection model—an empirical evaluation. *Computers*, 12(7):144, 2023.

14 Rosario Michel-Villarreal, Eliseo Vilalta-Perdomo, David Ernesto Salinas-Navarro, Ricardo Thierry-Aguilera, and Flor Silvestre Gerardou. Challenges and opportunities of generative ai for higher education as explained by chatgpt. *Education Sciences*, 13(9), 2023. doi:10.3390/educsci13090856.

15 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Fgpe authorkit–a tool for authoring gamified programming educational content. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*, pages 564–564, 2020.

16 José Carlos Paiva, Ricardo Queirós, José Paulo Leal, and Jakub Swacha. Yet another programming exercises interoperability language (short paper). In *9th Symposium on Languages, Applications and Technologies (SLATE 2020)*. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2020.

17 Ivica Pesovski, Ricardo Santos, Roberto Henriques, and Vladimir Trajkovik. Generative ai for customizable learning experiences. *Sustainability*, 16(7), 2024. doi:10.3390/su16073034.

18 Prajish Prasad and Aamod Sane. A self-regulated learning framework using generative ai and its application in cs educational intervention design. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, SIGCSE 2024, pages 1070–1076, New York, NY, USA, 2024. Association for Computing Machinery. doi:10.1145/3626252.3630828.

19 Junaid Qadir. Engineering education in the era of chatgpt: Promise and pitfalls of generative ai for education. In *2023 IEEE Global Engineering Education Conference (EDUCON)*, pages 1–9, 2023. doi:10.1109/EDUCON54358.2023.10125121.

20 Ricardo Alexandre Peixoto de Queirós. Integration of a learning playground into a lms. In *Proceedings of the 27th ACM Conference on on Innovation and Technology in Computer Science Education Vol. 2*, pages 626–626, 2022.

21 Laurie Ruettimann. How to interact with chat gpt-4 effectively: Top tips for better questions, September 2023. URL: https://laurieruettimann.com/chat-gpt-4-ask-questions/.

22 Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1*, ICER '22, pages 27–43, New York, NY, USA, 2022. Association for Computing Machinery. doi:10.1145/3501385.3543957.

23 Jiahong Su and Weipeng Yang. Unlocking the power of chatgpt: A framework for applying generative ai in education. *ECNU Review of Education*, 6:1–12, April 2023. doi:10.1177/20965311231168423.

24 John Wrenn, Shriram Krishnamurthi, and Kathi Fisler. Who tests the testers? In *Proceedings of the 2018 ACM Conference on International Computing Education Research*, ICER '18, pages 51–59, New York, NY, USA, 2018. Association for Computing Machinery. doi:10.1145/3230977.3230999.