ISSN 2529-9824

Artículo de Investigación

Language models for generating programming questions with varying difficulty levels

Modelos de lenguaje para la generación de preguntas de programación con diferentes niveles de dificultad

Christian Lopez[1]: Lafayette College, United States of America & Universidad Nacional Pedro Henríquez Ureña (UNPHU), Dominican Republic.
lopezbec@lafayette.edu
Miles Morrison: Lafayette College, United States of America.
morrismp@lafayette.edu
Matthew Deacon: Lafayette College, United States of America.
deaconmp@lafayette.edu

Date of Reception: 30/05/2024

Acceptance Date: 25/07/2024

Publication Date: 12/09/2024

How to cite the article:

Lopez, C., Morrison, M., & Deacon, M. (2024). Language models for generating programming questions with varying difficulty levels [Modelos de lenguaje para la generación de preguntas de programación con diferentes niveles de dificultad]. *European Public & Social Innovation Review*, 9, 1-19. https://doi.org/10.31637/epsir-2024-760

Abstract:
Introduction: This study explores the potential of Large Language Models (LLMs), specifically ChatGPT-4, in generating Python programming questions with varying degrees of difficulty. This ability could significantly enhance adaptive educational applications. **Methodology:** Experiments were conducted with ChatGPT-4 and participants to evaluate its ability to generate questions on various topics and difficulty levels in programming. **Results:** The results reveal a moderate positive correlation between the difficulty ratings assigned by ChatGPT-4 and the perceived difficulty ratings given by participants. ChatGPT-4 proves to be effective in generating questions that cover a wide range of difficulty levels.**Discussion:** The study highlights ChatGPT-4's potential for use in adaptive educational applications that accommodate different learning competencies and needs. **Conclusions:** This study presents a prototype of a gamified educational application for teaching Python, which uses ChatGPT to automatically generate questions of varying difficulty levels. Future studies should conduct

[1]**Autor Correspondiente**: Christian Lopez: Lafayette College (USA).

more exhaustive experiments, explore other programming languages, and address more complex programming concepts.

Keywords: Large Langue Models; ChatGPT; Question Generation; Adaptation; Gamification; Python; Difficulty; Pedagogy.

Resumen:
Introducción: Este estudio explora el potencial de los Modelos de Lenguaje Extenso (MLE), específicamente ChatGPT-4, en la generación de preguntas de programación en Python con diferentes grados de dificultad. Esta capacidad puede mejorar las aplicaciones educativas adaptativas. **Metodología:** Se realizaron experimentos con ChatGPT-4 y participantes para evaluar su capacidad de generar preguntas sobre diversos temas y niveles de dificultad en programación. **Resultados:** Los resultados revelan una correlación positiva moderada entre las clasificaciones de dificultad asignadas por ChatGPT-4 y las calificaciones de dificultad percibida por los participantes. ChatGPT-4 demuestra ser eficaz en la generación de preguntas de distintos niveles de dificultad. **Discusión:** El estudio destaca el potencial de ChatGPT-4 para ser utilizado en aplicaciones educativas adaptativas que se ajusten a las diferentes competencias y necesidades de los estudiantes. **Conclusiones:** Este estudio presenta un prototipo de una aplicación educativa gamificada para enseñar Python, que utiliza ChatGPT para generar preguntas automáticamente. Se sugiere que futuros estudios realicen experimentos más exhaustivos, exploren otros lenguajes de programación y aborden conceptos más complejos.

Palabras clave: Modelos de Lenguaje Extenso; ChatGPT; Generación de Preguntas; Adaptación; Gamificación; Python; Dificultad; Pedagogía.

1. Introduction

Programming is an invaluable skill that opens career opportunities, develops problem-solving skills, and enhances technical literacy (Scherer *et al.*, 2021). However, teaching and engaging students with programming can be challenging (Sinclair *et al.*, 2015). Programming requires a unique logical thinking style, which often takes time for students to acquire (Albán-Bedoya & Ocaña-Garzón, 202). Additionally, many programming constructs are highly abstract, making them difficult to communicate effectively. Two popular solutions to try and mitigate these issues are introducing students to programming through high-level programming languages and leveraging gamification.

Gamification has gained attention as a means of increasing student engagement (Saleem *et al.*, 2022; Oliveira *et al.*, 2023). Gamification is the design methodology of implementing elements or mechanics usually present in game (e.g., points, levels) into non-game domains to enhance user experience and elicit certain behaviors, like engagement (Deterding *et al.*, 2011; Huotari & Hamari, 2017). Engagement is a common concern in education due to its relationship with students' academic achievements (Lei *et al.*, 2018). This is especially relevant in computer science which rates lower than average on student engagement benchmarks compared to other subjects (Sinclair *et al.*, 2015). Consequently, research has explored the use of gamification applications in the context of computer science and teaching programming languages (Ahmad *et al.*, 2020; Zhan *et al.*, 2022). However, gamification research highlights the importance for advancing gamification systems capable of adapting to unique student characteristics to keep them engaged for a longer period (Bennani *et al.*, 2022). For example, automating question generation to accommodate varying student proficiencies would be an invaluable resource that could enhance the learning experiences (Sarsa *et al.*, 2022). Thankfully, recent advancements in Artificial Intelligence (AI), such as Language Models, have facilitated the generation of questions for educational applications.

1.1. Large Language Models

Large Language Models (LLMs) are computational models designed to manipulate, generate and understand human language (Chang *et al.*, 2024). The size of LLMs is usually based on the number of parameters it has, which significantly correlates with their performance (Caruccio *et al.*, 2024; Gemini Team *et al.*, 2023; OpenAI *et al.*, 2023). OpenAI's ChatGPT, is a great example of the relationship of parameters and performance since GPT-4, which is estimated to have trillions - exact numbers are not publicly disclosed - significantly outperforms GPT-3 on multiple benchmark datasets, which had 175 billion parameters (OpenAI *et al.*, 2023). These advancements in LLMs have opened new avenues for research in educational contexts.

The applications of LLMs in education are extensive (Wang *et al.*, 2024). Three categories – *Study Assisting*, *Teach Assisting*, and *Adaptive Learning* – were described by Wang and colleagues in their taxonomy of LLMs uses in education. Study Assisting involves LLMs aiding students directly, Teach Assisting involves aiding teachers, and Adaptive Learning involves LLMs automating and personalizing parts of the learning process. These three categories are further subdivided: Study Assisting included functionality of: (i) Question Solving, (ii) Error Correction, and (iii) Confusion Helper. Teaching Assisting included functionality of: (i) Question Generation, (ii)Automatic Grading, and (iii) Material Creation. Lastly, Adaptive learning included functionality of: (i) Knowledge Tracing and (ii) Content personalization (Wang *et al.*, 2024).

The use of LLMs for question generation in educational contexts has garnered significant attention by researchers (Biancini *et al.*, 2024; Doughty *et al.*, 2024; Zhang *et al.*, 2022). For example, Biancini *et al.* (2024) tested three LLMs for their ability to generate multiple choice questions (MCQs). They found that ChatGPT was particularly proficient at generating MCQs and produced the highest quality among the LLMs they evaluated. Their study also suggests integrating user personalization characteristics into LLM prompts to further improve the quality of questions produced and enhance the student experience (Biancini *et al.*, 2024).

Several studies emphasize that the quality of prompts provided as input to LLMs significantly affects the quality of the generated output. Various prompt engineering methodologies have been described in literature. Multiple studies have identified key strategies, such as: (i) using clear and precise language, (ii) specifying the role the LLM should assume (e.g., student, teacher, expert, writer), (iii) encouraging the model to operate step-by-step and (iv) defining the desired output format (Amatriain, 2024; Chen *et al.*, 2023; Ortolan, 2023; Velasquez-Hainao *et al.*, 2023; Zhou *et al.*, 2022)

1.2. Large Language Models for Programming

There are many top performing LLMs on the market currently, including ChatGPT, Gemini, and Claude (Caruccio *et al.*, 2024; Gemini Team *et al.*, 2023; OpenAI *et al.*, 2023). The performance of LLM is often evaluated through specialized benchmarks designed to test the model's capability in specific Natural Lnauge Processing (NLP) tasks (Liu *et al.*, 2023). One NLP tasks that has gained the attention of researchers is programing and/or coding tasks (Hou *et al.*, 2023). For example, HumanEval+, a dataset composed of publicly available code from GitHub used to study Python code-writing capabilities, has been widely used for comparing LLMs (M. Chen *et al.*, 2021; Liu *et al.*, 2023). As of May 13, 2024, ChatGPT4o is the state-of-the-art LLM on this benchmark (i.e., it performs the best) (Liu *et al.*, 2023).

Many researchers are exploring the use of LLMs in teaching programming languages like Python. For example, Sarsa *et al.*, (2022) tested LLMs for their ability to generate programming exercises and code explanations, finding that they could create novel and sensible questions. They emphasized the need for additional oversight because of potential output inaccuracies. This is one of the reasons why a two-step process is often recommended, where an LLM also validates the output and format of another LLM (Shankar *et al.*, 2024). Similarly, Doughty et. (2024) al tested ChatGPT-4 on its capacity to generate MCQs about Python. They found that Python programming MCQs generated by ChatGPT-4 were comparable to human-crafted questions and, in some cases, had better alignment with learning objectives. However, they identified a few problems with the question generation, such as MCQs with multiple correct answers. While their study supports the capability of LLMs, specifically ChatGPT-4, to generate python programming questions, it did not explore the difficulty of the generated MCQs (Doughty *et al.*, 2024).

1.3. Questions Difficulty

Recent studies indicate that researchers are increasingly interested in exploring how LLMs can be leveraged in educational contexts to generate a diverse set of questions. However, there is still a lack of understanding of the nuanced capacity of LLMs to generate questions of varying difficulty (Doughty *et al.*, 2024). Complexity, difficulty and challenge are distinct terms that can mean very different things depending on the context. Difficulty and challenge are often thought of as a user's response to complexity. Thus, defining complexity and attempting to quantify it can help predict difficulty and challenge. In education, complexity often considers the structural characteristics of a task, such as the number of components, the interactivity between components, and the cognitive load associated with each component (Chen *et al.*, 2023).

Many approaches have been taken to describe complexity and its subparts in a meaningful way. For example, Blooms Taxonomy's is widely used in pedagogy and education because its cognitive processes correlate effectively with complexity and cognitive load. (Krathwohl, 2002). Similarly, the Cognitive Load Theory, a theoretical framework that distinguishes between sources of cognitive effort in a task or subtask, has also been utilized (Sweller, 1988). Sweller (1988) describes three types of cognitive load – Intrinsic cognitive load, extrinsic cognitive load, and germane cognitive load. Intrinsic cognitive load is the inherent mental effort associated with a task. Extrinsic cognitive load refers to the mental effort associated with how a task is presented. Lastly, Germane cognitive load refers to the effort associated with processing, constructing and automating schemes. Each of the Cognitive Load Theory constructs have substantial subjective characteristics. There are some ways to try objectively quantifying each – textual characteristics like word count, usage and format contribute to extrinsic load and user response time captures cognitive load generally to some extent. However, cognitive load and its subclass ultimately depend on a user, making them inherently subjective (Zu *et al.*, 2021). Identifying and managing complexity in education is crucial for creating effective learning environments that cater to the needs of all students. Providing learning materials with an appropriate challenge level, tailored to the spectrum of student proficiencies, can significantly promote student engagement, motivation, focus, and foster a mental state of flow (Flegal *et al.*, 2019; Yazidi *et al.*, 2020)

In the context of programming education, many metrics are used to quantify the complexity of code. Metrics like Halstead and McCabe (cyclometric) complexity consider factors such as the number of operators & operands or the control flow of a program, respectively. There are also simpler metrics to describe code complexity like Lines of Code and syntax-based concept counts. Interestingly, basic metrics of code complexity perform better in tracking complexity for introductory topics than their more sophisticated alternatives (Ihantola & Petersen, 2019).

The implementation of LLMs, like ChatGPT, into programming education presents a promising avenue for enhancing student engagement and learning outcomes. LLMs could offer new capabilities for adapting educational content, particularly through question generation, which can be tailored to match students' varying skill levels. Nevertheless, understanding and quantifying the difficulty of generated questions remains essential for creating effective adaptive learning systems. As research progresses, the use of LLMs could revolutionize the way programming is taught, making it more accessible and engaging for students at all levels. Towards this end, this work conducts a series of experiments to explore the potential of LLMs, specifically ChatGPT-4o, in generating Python programming questions with varying degrees of difficulty. Lastly, this work also introduces a prototype of a gamified educational application design to teach Python programming that leverages ChatGPT to automatically generate challenges (i.e., questions) of different levels of difficulty.

2. Method

A set of experiments were conducted to explore the capability of LLMs to generate Python programming questions of different degrees of difficulty. OpenAI ChatGPT-4o was chosen due to its top performance in coding tasks (see section 1.2). To assess its ability, both human participants and ChatGPT-4o itself were employed to assess the difficulty of the generated questions.

2.1. *Questions generation using ChatGPT-4o*

The generation process of the python programming questions utilized OpenAI Application Programming Interface (API) for automation. Specifically, the System prompt and User prompt shown below were used in each of the calls made to the API to generate a corpus of python programming questions that would be further analyzed.

System prompt:
"Core Block
You are an expert Python question generator focused on creating high-quality questions of varying difficulty to teach students. The user prompt will specify the type of question (multiple-choice, fill-in-the-blank, true/false, drag-and-drop), relative difficulty level (1-10, where 1 is the easiest), and question topic (loops, data types, variables, syntax, indexing, etc.). Diversify the question subtypes or methods used within a particular question category. Ensure each part has only one unambiguous correct answer. Maintain a high standard of question quality, ensuring clarity, precision, and proper grammar.

Verification Block
1. Verify the correct answer is the correct answer. If not, replace it with the correct answer. Repeat until the answer passes your evaluation.
2. Verify each incorrect answer is incorrect. If correct, replace it with an incorrect answer. Repeat until the answer
passes your evaluation.
3. Verify the functionality of code blocks.
4. Verify that the question difficulty matches the requested difficulty level.
5. Verify the diversification of question subtypes or methods."

The *System prompt* had two main blocks. The "*Core Block*" assigns ChatGPT its role and outlined the types of questions it would generate, while the "*Verification Block*" ensured that ChatGPT executes the "*Core Block*" properly. The role assigned to the question generator in the first sentence of the *"Core Block"* was inspired by the prompt used by Doughty. *et al.* (2024). Additionally, other existing work supports the idea of providing a role description for the LLM at the beginning of prompts. Furthermore, Lee *et al.*, (2023) highlighted difficulties in differentiating subcategories and produced questions of the same type. This information influenced the decision to command the prompt to diversify the question subtypes. Moreover, the work presented in Doughty *et al.*, (2024) supports the idea to command the model, via the prompt, to have one ambiguous answer and promote a high-quality standard for each question. Lastly, Shin & Ramanathan (2023) found that when the prompt requested to provide every step of a calculation for math problems, ChatGPT's accuracy improved. A similar ideology was applied when creating the verification block, as adding extra steps in to ensure correctness, difficulty, and diversification lead to more ideal results.

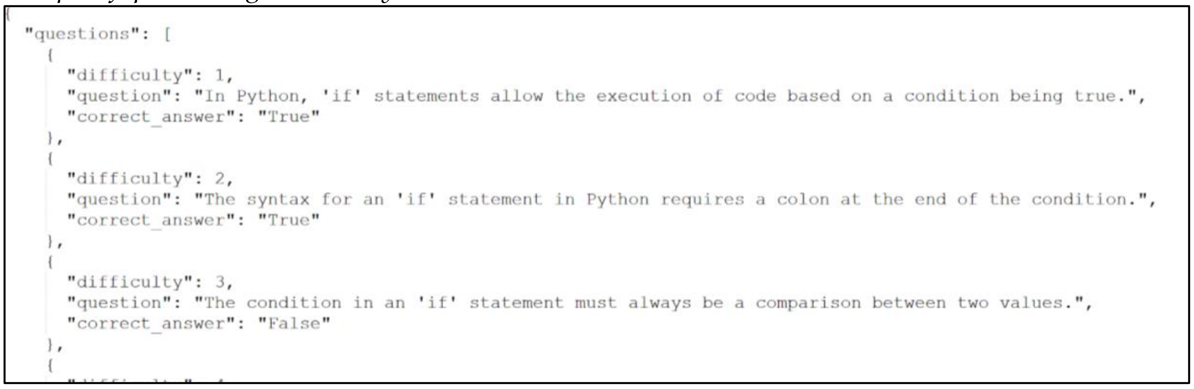
Subsequently, the following *User prompt* was used to specify the type of question and topic it needed to generate: "*Create 10 {question_type} questions about {topic}. Where each {question_type} question increments by 1 in difficulty, going from 1 to 10. Respond with the questions formatted as JSON objects*."

The *User prompt* was modified automatically based on the required question type and topic. Question types alternated between: (i) Multiple-Choice questions, (ii) True/False questions, (iii) Drag-and-Drop, and (iv) Fill-in-the-Blank. Similarly, the topic altered between: (i) If-statements and (ii) Loops. Hence a prompt like "*Create 10 Multiple-Choice questions about If-statement. Where each Multiple-Choice question increments by 1 in difficulty, going from 1 to 10. Respond with the questions formatted as JSON object"* would have generated 10 Multiple-Choice questions focused on Python If-statement constructs. For each combination of questions type and topic, ten API calls were executed. This was done with the purpose of exploring some of the generation's randomness (e.g., creativity) of ChatGPT-4o since the "*temperature'* hyperparameter of the model was set to 1 (*API Reference - OpenAI API*; Davis *et al.*, 2023; Ekin, 2023; Shieh J., 2023). Therefore, a total of 80 sets of 10 questions of different levels of difficulty ranging from 1 to 10, were generated.

The *User prompt* and API call parameters required ChatGPT-4o to output responses in a JSON format. This was done to help with the subsequent analysis presented in this work. Moreover, it was done to explore the feasibility of integrating this prompt system into a gamified educational programming application (see section 3.3). The JSON files not only contained the questions and their correct answers but also the difficulty level. For example, Figure 1 shows part of the output for ChatGPT when prompted to generate ten True/False questions about if-statements. Lastly, the total number of lines of code as well as the total number of characters contained in each of the questions (i.e., length) were calculated for subsequent analysis. This was done with the objective of exploring if there was any correlation between the difficulty level of the questions and the number of lines of code the questions used, or the length of the questions itself.

Figure 1.

Example of questions generated by ChatGPT-4o



```
{
  "questions": [
    {
      "difficulty": 1,
      "question": "In Python, 'if' statements allow the execution of code based on a condition being true.",
      "correct_answer": "True"
    },
    {
      "difficulty": 2,
      "question": "The syntax for an 'if' statement in Python requires a colon at the end of the condition.",
      "correct_answer": "True"
    },
    {
      "difficulty": 3,
      "question": "The condition in an 'if' statement must always be a comparison between two values.",
      "correct_answer": "False"
    },
    {
```

2.2. *Evaluations of questions using ChatGPT-4o*

Prior to implementing ChatGPT-4o to rank the questions generated based on their difficulty level, each group of ten questions was stripped of any difficulty related information and then randomized (i.e. scrambled). This was done to prevent ChatGPT-4o from picking up on the pattern of increasing difficulty. The original rankings (difficulty level) provided in step 2.1 were stored for later comparison. ChatGPT-4o API was used to iterate through each set of questions to rank them based on their relative difficulty. The following *System prompt* and *User prompt* were used to achieve the ranking of the question:

System Prompt

"You are an expert Python question evaluator focused on evaluating high-quality questions of varying difficulty to teach students. The user prompt will specify the type of questions (multiple-choice, fill-in-the-blank, true/false, drag-and-drop) to evaluate, and the question topic (loops or if statement). You will rank the questions based on their relative difficulty level of each of the questions from 1-10, where 1 is the easiest. You will ensure that:

-All questions have a ranking associated with them
-No two questions have the same ranking
-All rankings are based on the difficulty level of the question
-The simplest question is rank 1
-The most difficult question is rank 10
-All questions rank between 1 and 10, 1 being the simplest and 10 the most difficult.
-If your responses contain text, try again until your response is only numbers

You will provide your answers as a sequence of ranking numbers following the order in which the questions were given. For example, the output: [3,6,7,10,8,9,2,4,5,1] will mean that the first question given in the user prompt was ranked as the 3rd question based on difficulty level, the second question as the 6th question based on difficulty level, and so on."

User Prompt
"Rank these 10 {question_type} about {question_topic}. Your response has to be as a sequence of ranking numbers following the order in which the questions were given. The questions are: {questions}"

The style of the system prompt is modeled after the question generation prompt to maintain consistency and utilize similar principles found in the literature (see sections 2.1). For instance, ChatGPT-4o is given a role and there is automatic verification of the output. To ensure the output of ChatGPT-4o conformed to the output needed for the subsequent analyses (i.e., a list of 10 numbers from 1 -10 without repetition), a verification script was also used. If the output format was incorrect, a new API call was executed for that set of questions. Each set of ten questions was ranked ten times to observe the variation in ChatGPT's rankings of the questions.

2.3. Evaluations of questions using human participants

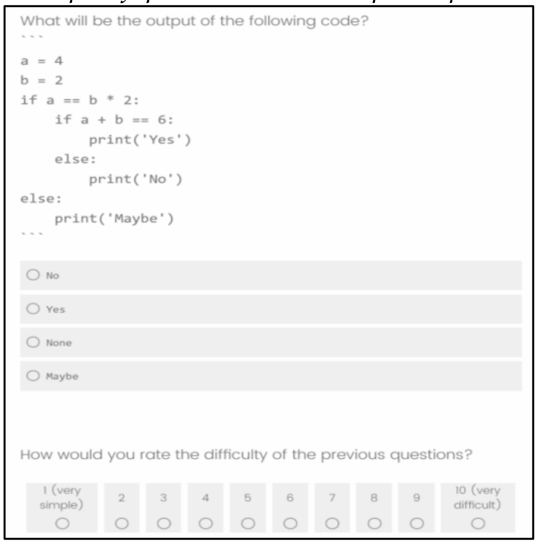
To assess ChatGPT-4o's ability to generate Python programming questions of different degrees of difficulty, a human-subject experiment was completed. Participants were asked to answer a set of ten questions and rate the difficulty level of each question. Via a survey, participants were asked to rate the questions instead of ranking them to minimize the required cognitive load necessary to complete the survey (e.g., for ranking, all ten questions need to be assessed together, while for rating only one at a time). Before answering the questions, volunteers were asked to complete a consent form where they agreed to allow their responses to be shared anonymously. They were then asked to indicate their level of experience programming in Python, as well as how many years they had used Python.

For this experiment, two sets of If-statements Multiple-choice questions were randomly selected. Choosing only one topic helped keep the experiment size manageable. Moreover, If-statements was chosen instead of Loops because of their relative simplicity (e.g., to implement loops requires knowledge of control statements - if-statements) (Gomes *et al.*, 2019). Moreover, multiple choice questions were chosen because they had more answer options than True/False questions, decreasing the likelihood of correct guesses. Similarly, this type of question can only have one correct answer option, unlike Fill in the Blanks questions, which make them easy to implement and evaluate in a survey.

The questions from the randomly chosen sets were presented in random order to the participants. Participants were only exposed to ten questions from the same set. The set to which participants were exposed was also random. Choices for each question were also randomly ordered, except for cases where options like "All of the above" or "None of the above" were applicable. After answering each question, participants rated the difficulty of the questions with a ten-point Likert scale with anchors at one (1- very simple) and ten (10- very difficult). Response times for each question pair were measured during the survey. Figure 2 presents a visual of the screen displayed to participants.

Figure 2.

Example of questions shown to participants



What will be the output of the following code?

```
```
a = 4
b = 2
if a == b * 2:
    if a + b == 6:
        print('Yes')
    else:
        print('No')
else:
    print('Maybe')
```
```

How would you rate the difficulty of the previous questions?

Lastly, the survey included two control questions randomly mixed in with the questions of interest. The purpose of the control question was to ensure that the participants were reading the questions and not just randomly selecting answers. The control question asks the participant to choose a specific answer without assessing any knowledge of the content.

3. Results

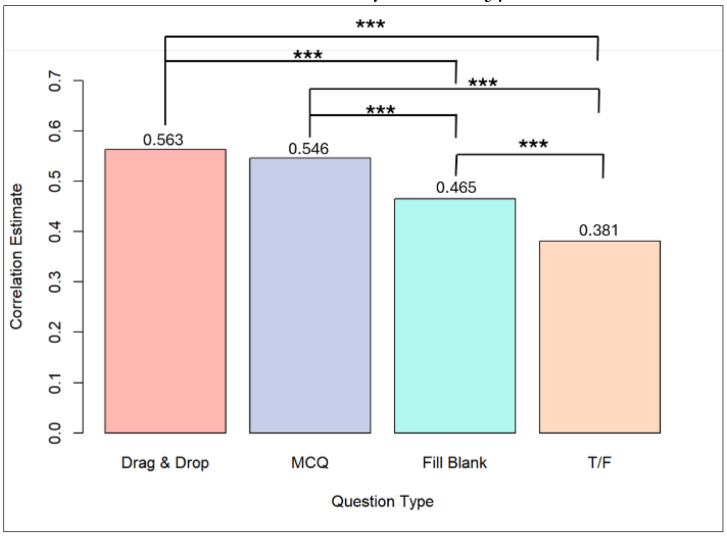
3.1. Evaluation of questions using ChatGPT-4o

As introduced in section 2, ChatGPT-4o was implemented to rank the questions generated (see sections 2.1, and 2.2). A Spearman's rank correlation test was conducted to assess the relationship between the original rankings (see section 2.1) and new rankings (see section 2.2). There was a moderate positive correlation between the two rankings ($\rho = 0.488,\ p - value < 0.001$). This result suggests that there is a statistically significant association between the original and new rankings. Hence, the null hypothesis that there is no correlation was rejected, supporting the alternative hypothesis that the true rho is not equal to zero. All this suggests that while the new and original ranking were not the same, there were some significant agreements.

Furthermore, to explore if the questions type and the topics had any confounding effects on the correlation estimated between the original and new ranking, Fisher's transformation for the correlation coefficients were performed. As shown in Figure 3, the test for the confounding effects of the questions type shows that the strongest correlation was for Drag-and-Drop questions, followed by Multiple-Choice-Questions (MCQ). The weakest correlation was for True and False questions (T/F). Moreover, all the pairwise correlations comparison tests, except for "Drag-and-Drop vs MCQ", show significant statistical difference. Similarly, when looking at the correlations confounded by topic, results show that for the "If-statements" ($\rho = 0.512$) the correlations where greater than for the "Loop" questions ($\rho = 0.465$). Moreover, the Fisher's transformation for the correlation coefficients show that this difference was statistically significant ($p - value < 0.01$). Lastly, the spearman's rank correlation tests between the rankings and the number of lines of code of the questions show very weak positive correlations ($original\ ranking$: $\rho = 0.119,\ p - value < 0.001$, new ranking: $= 0.085$, $p - value < 0.001$). Similarly, the spearman's rank correlation tests between the rankings and the number of characters of the questions (i.e., length) show weak positive correlations ($original\ ranking$: $\rho = 0.286,\ p - value < 0.001$, new ranking: $= 0.231,\ p - value < 0.001$).

Figure 3.

Correlation estimates based on question types



****p-value<0.001*

3.2. Evaluation of questions using human participants

For this work, participants were recruited via Amazon Mechanical Turk (AMT) [https://www.mturk.com/]. AMT offers a low-cost access to a diverse pool of participants, and it has been used in human-subject experiments (Aguinis *et al.*, 2021), and specifically experiment to assess complexity in the context of LLM (Busheska & Lopez, 2022; Mcshane & Lopez, 2023). For taking the time to voluntarily be part of this study, participants were compensated a total of $4.20 (USD) upon completing the questionnaires and correctly answering quality control questions.

All the responses collected were filtered based on the conditions that: (i) the participants complete all the questions, (ii) they passed the two quality control questions, and (iii) they correctly responded the questions with difficulty level 1 and 2. This was done to help ensure that analyzed responses came from participants who demonstrated basic understanding of Python programming rather than random clicking. After all this filtering, a total of 25 participants' responses were analyzed. On average these participants took 13 minutes to complete the survey (Mn=11.8, Min=7.7, Max=37.5, SD=6.2). They reported having on average 2.12 years of experience with Python programming (Mn=2, Min=1, Max=5, SD=1,6).

A Spearman's rank correlation test was conducted to assess the relationship between the original ranking generated by ChatGPT-4o and the rating of the human participants. The analysis yielded a Spearman's rho value of 0.290 (p-value<0.001), indicating a weak positive correlation between the two sets of scores that was statistically significant. Moreover, when looking at the average ranking generated by ChatGPT between the questions that were correctly answered by participants (μ= 0.85) and those questions that were not correctly answered (μ= 5.50), a two-sample t-test showed a statistically significant difference between the average ranking of the two groups (p-value<0.001). Lastly, there were no statistically significant correlations between the rankings and the number of lines of code or the length of the questions. This could be attributed to the low sample size.

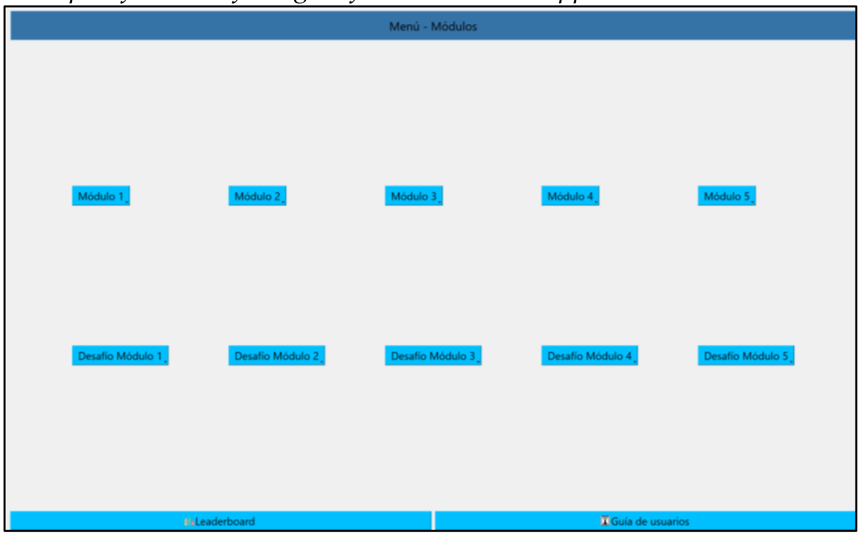
3.3. Integration into gamified educational application

This work also explores the integration of ChatGPT into a gamified educational application with the goal of automatically generating questions of different levels of difficulty. Specifically, a script pipeline that leverages ChatGPT API and the prompts shown in section 2.1 was integrated into the prototype of an application design to teach Python programming for Spanish-speaking users.

The gamified application features several game elements, such as points, leaderboards, badges, and challenges. It is divided into five teaching modules covering topics like variables, if statements, loops, lists, and strings. After completing each module, users face a "challenge" consisting of a series of questions related to the module's topic (see Figure 4, where "Desafío" means "challenge" in Spanish). These challenges include various question types, such as True-False, Drag-and-Drop, and Multiple Choice Questions (MCQs), all of which are automatically generated using the ChatGPT API. The application's code and further details can be found here: [https://github.com/lopezbec/AI_Gamification_Python].
To render different types of questions in the User Interface (UI), the application utilizes JSON files of a specific format. Therefore, the prompt shown in section 2.1 was updated to generate JSON files in the required format for each question type. Additionally, a validation script was integrated to ensure that the API outputs JSON files in the expected format. Finally, a subsequent ChatGPT API call was employed to translate the selected questions into Spanish. Since most of the training data for ChatGPT in coding tasks is in English, it was deemed more effective to generate programming questions in English first and then translate them into Spanish.

Figure 4.

Example of the UI of the gamified educational application



4. Discussion

The primary aim of this study was to assess the capability of ChatGPT-4o to generate Python programming questions of varying difficulty levels. To achieve this, a corpus of questions with varying degrees of difficulty were generated and two sets of experiments conducted with them. The results demonstrate ChatGPT-4o's consistency in its internal model of what factor makes Python programming questions difficult, evidenced by the moderate positive correlation between the original difficulty rankings and the new rankings assigned by the model. This significant correlation suggests that ChatGPT-4o's internal difficulty ranking aligns with the intended difficulty levels to a considerable extent, helping support the model's efficacy in generating appropriately challenging questions.

An analysis of different question types revealed nuanced performance by ChatGPT-4o. The strongest correlation for Drag-and-Drop questions, followed by Multiple-Choice Questions, indicates that the model is better able at generating and ranking questions of wider range of difficulty levels. The weaker correlation for True and False questions suggests that while ChatGPT-4o can handle simple binary choices, its differentiation of subtle difficulty variations in such questions might be less effective. This variance aligns with educational theory, which posits that questions requiring higher-order thinking skills (e.g., Drag-and-Drop) are more complex, which could explain why ChatGPT-4o does a better job discriminating their difficulty level (Jones *et al.*, 2009). Topic-based analysis further supports the model's nuanced capabilities. Questions related to “If-statements” had a higher correlation compared to “Loop” questions, indicating that ChatGPT-4o might be more adept at generating and ranking questions on simpler programming constructs. Lastly, the moderate correlations between the rankings and the number of characters suggest that the length of the question plays a more substantial role in the ranking decisions compared to the number of lines of code. This could be due to the possibility that longer questions might be perceived as more detailed or comprehensive, thereby earning higher rankings. However, the fact that only weak correlations were found indicates that other factors could play a more important role in the ranking of the questions.

The human subject experiment provided additional insights into the model's capabilities and the difficulty validity of its generated questions. Despite its weakness, the significance of the correlation suggests some alignment between difficulty levels of the questions generated and human perceptions of question difficulty. Further analysis showed that questions correctly answered by participants had a significantly lower average ranking compared to questions they did not answer correctly. This suggests that ChatGPT-4o effectively generated difficult questions, as these were the ones participants struggled with more. This finding underscores the model’s ability to create questions that genuinely reflect varying levels of difficulty, as perceived by human participants.

The ability of ChatGPT-4o to generate questions of varying difficulty has significant practical implications. For educators and instructional designers, this capability can streamline the creation of diverse assessment materials tailored to different learning stages. This capability could facilitate the integration of LLM pipelines into educational applications that adapt content to student skill levels. However, as demonstrated in this study, integrating LLMs into educational applications requires implementing validation steps to ensure the generated questions comply with the format needed for the educational application to render the questions in the UI. Moreover, as suggested by Wang *et al.* (2024), achieving an adaptive learning experience requires not only content personalization but also knowledge tracing. Lastly, the findings of this work could help support the idea that advanced models like ChatGPT-4o can manage complex, pedagogically sound question generation and capture nuanced factors that contribute to the perceived difficulty of programming questions.

4.1. Limitations and Future Research

Despite the encouraging results, the study has limitations that warrant discussion. The moderate correlation between the original and new rankings suggests there is still room for improvement in ChatGPT-4o's ranking accuracy. Future research should focus on refining the model's algorithms to enhance its ability to differentiate between subtle gradations of difficulty, particularly for simpler question types like True-and-False.

The reliance on Amazon Mechanical Turk for participant recruitment, while offering a broad participant base, introduces variability in the responses due to diverse backgrounds and programming experience. Future studies should aim for a larger and a more controlled participant selection to ensure a more homogenous sample in terms of Python programming expertise, thus providing a clearer assessment of the model's capabilities.

Additionally, the human-subject experiment focuses on a specific question type and topic, which limits the generalizability of the findings. Future research should explore a wider range of programming constructs and question formats to provide a comprehensive understanding of ChatGPT-4o's capabilities. Investigating the model's performance in generating questions on more advanced topics or across different programming languages would also be valuable.

5. Conclusions

This study explores the potential of Large Language Models (LLMs), specifically ChatGPT-4o, in generating Python programming questions with varying degrees of difficulty. Programming is an invaluable skill that opens career opportunities, develops problem-solving skills, and enhances technical literacy. However, teaching and engaging students with programming can be challenging. Educators have employed several approaches to try and mitigate these issues, such as gamification. Unfortunately, these efforts appear inadequate -- research has indicated the need to move toward adaptive systems capable of tailoring its content (e.g., questions, tasks) to unique users' skill levels. Thankfully with the advancement of LLMs, researchers have started exploring how they can be used for automatic questions generation.

The ability to automatically create questions of different difficulties could significantly enhance adaptive educational applications. Towards this end, a series of experiments involving both ChatGPT-4o and human participants were conducted to evaluate ChatGPT-4o's capacity to generate diverse question types across various topics and difficulty levels. The findings reveal a weak positive correlation between the difficulty rankings assigned by ChatGPT-4o and the perceived difficulty ratings given by participants. These results indicate that LLMs, like ChatGPT-4o, could effectively generate questions that span a wide range of difficulty levels.

Moreover, integrating a LLM pipeline into an existing gamified educational application helps showcases challenges educators might face when leveraging LLMs. Nevertheless, such integration could allow for adaptive learning experiences where content is dynamically tailored to student skill levels, potentially enhancing engagement and educational outcomes. By combining game elements like points, leaderboards, badges, and challenges with the automated generation of questions of different level of difficulty, the learning environment could become more interactive and motivating for students. Though, by addressing the limitations of this work and expanding the scope of future research, the reliability and applicability of AI-driven question generation can be further enhanced. This would ultimately contribute to more efficient and effective educational applications, transforming learning experiences to be more personalized, engaging, and effective.

6. References

Aguinis, H., Villamor, I., & Ramani, R. S. (2021). MTurk Research: Review and Recommendations. *Journal of Management*, 47(4), 823–837. SAGE Publications Inc. https://doi.org/10.1177/0149206320969787

Ahmad, A., Zeshan, F., Khan, M. S., Marriam, R., Ali, A., & Samreen, A. (2020). The Impact of Gamification on Learning Outcomes of Computer Science Majors. *ACM Transactions on Computing Education*, *20*(2). https://doi.org/10.1145/3383456

Albán Bedoya, I., & Ocaña-Garzón, M. (2022). Educational Programming as a Strategy for the Development of Logical-Mathematical Thinking. *Lecture Notes in Networks and Systems*, 405 LNNS, 309–323. https://doi.org/10.1007/978-3-030-96043-8_24

Amatriain, X. (2024). *Prompt Design and Engineering: Introduction and Advanced Methods*. 1–26. http://arxiv.org/abs/2401.14423

Amazon. (2018). *Amazon Mechanical Turk*. https://www.mturk.com/

API Reference - OpenAI API. Retrieved December 10, 2023, from https://platform.openai.com/docs/api-reference/chat

Baudisch, P., Beaudouin-Lafon, M., Mackay, W., Association for Computing Machinery, SIGCHI (Group: U.S.), & ACM Digital Library. (2013). *CHI2013 Changing perspectives : extended abstracts : the 31st Annual CHI Conference on Human Factors in Computing Systems : 27 April - 2 May, 2013, Paris, France.*

Bennani, S., Maalel, A., & Ben Ghezala, H. (2022). Adaptive gamification in E-learning: A literature review and future challenges. *Computer Applications in Engineering Education*, 30 (2), 628–642. https://doi.org/10.1002/cae.22477

Biancini, G., Ferrato, A., & Limongelli, C. (2024). Multiple-Choice Question Generation Using Large Language Models: Methodology and Educator Insights. *Adjunct Proceedings of the 32nd ACM Conference on User Modeling, Adaptation and Personalization*, 584–590. https://doi.org/10.1145/3631700.3665233

Busheska, A., & Lopez, C. (2022). Exploring the perceived complexity of 3d shapes: towards a spatial visualization VR application. *Proceedings of the IDETC-CIE 2022*, 1–9.

Caruccio, L., Cirillo, S., Polese, G., Solimando, G., Sundaramurthy, S., & Tortora, G. (2024). Claude 2.0 large language model: Tackling a real-world classification problem with a new iterative prompt engineering approach. *Intelligent Systems with Applications*, 21. https://doi.org/10.1016/j.iswa.2024.200336

Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., Ye, W., Zhang, Y., Chang, Y., Yu, P. S., Yang, Q., & Xie, X. (2024). A Survey on Evaluation of Large Language Models. *ACM Trans. Intell. Syst. Technol.*, *15*(3). https://doi.org/10.1145/3641289

Chen, B., Zhang, Z., Langrené, N., & Zhu, S. (2023). *Unleashing the potential of prompt engineering in Large Language Models: a comprehensive review*. http://arxiv.org/abs/2310.14735

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. de O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., … Zaremba, W. (2021). *Evaluating Large Language Models Trained on Code*. http://arxiv.org/abs/2107.03374

Chen, O., Paas, F., & Sweller, J. (2023). A Cognitive Load Theory Approach to Defining and Measuring Task Complexity Through Element Interactivity. *Educational Psychology Review*, 35 (2). https://doi.org/10.1007/s10648-023-09782-w

Davis, J., Van Bulck, L., Durieux, B., & Lindvall, C. (2023). The temperature feature of ChatGPT: Modifying creativity for clinical research (Preprint). *JMIR Human Factors*, 11. https://doi.org/10.2196/53559

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: Defining "gamification." *Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, MindTrek 2011*, 9–15. https://doi.org/10.1145/2181037.2181040

Doughty, J., Wan, Z., Bompelli, A., Qayum, J., Wang, T., Zhang, J., Zheng, Y., Doyle, A., Sridhar, P., Agarwal, A., Bogart, C., Keylor, E., Kultur, C., Savelka, J., & Sakr, M. (2024). A Comparative Study of AI-Generated (GPT-4) and Human-crafted MCQs in Programming Education. *ACM International Conference Proceeding Series*, 114–123. https://doi.org/10.1145/3636243.3636256

Ekin, S. (2023). *Prompt Engineering For ChatGPT: A Quick Guide To Techniques, Tips, And Best Practices*. https://doi.org/10.36227/techrxiv.22683919

Flegal, K. E., Ragland, J. D., & Ranganath, C. (2019). Adaptive task difficulty influences neural plasticity and transfer of training. *NeuroImage*, *188*, 111–121. https://doi.org/https://doi.org/10.1016/j.neuroimage.2018.12.003

Gemini Team, Anil, R., Borgeaud, S., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., Millican, K., Silver, D., Johnson, M., Antonoglou, I., Schrittwieser, J., Glaese, A., Chen, J., Pitler, E., Lillicrap, T., Lazaridou, A., … Vinyals, O. (2023). *Gemini: A Family of Highly Capable Multimodal Models*. http://arxiv.org/abs/2312.11805

Gomes, A., Ke, W., Lam, C. T., Teixeira, A., Correia, F., Marcelino, M., & Mendes, A. (2019). Understanding loops: a visual methodology. *2019 IEEE International Conference on Engineering, Technology and Education (TALE)*, 1–7. https://doi.org/10.1109/TALE48000.2019.9225951

Hou, X., Zhao, Y., Liu, Y., Yang, Z., Wang, K., Li, L., Luo, X., Lo, D., Grundy, J., & Wang, H. (2023). *Large Language Models for Software Engineering: A Systematic Literature Review*. http://arxiv.org/abs/2308.10620

Huotari, K., & Hamari, J. (2017). A definition for gamification: anchoring gamification in the service marketing literature. *Electronic Markets*, *27*(1), 21–31. https://doi.org/10.1007/s12525-015-0212-z

Ihantola, P., & Petersen, A. (2019). Code Complexity in Introductory Programming Courses. *Proceedings of the 52nd Hawaii International Conference on System Sciences*, 1–9. https://hdl.handle.net/10125/60204

Jones, K., Harland, J., Reid, J., & Bartlett, R. (2009). Relationship between examination questions and bloom's taxonomy. *Proceedings - Frontiers in Education Conference*, 1–6. https://doi.org/10.1109/FIE.2009.5350598

Lee, U., Jung, H., Jeon, Y., Sohn, Y., Hwang, W., Moon, J., & Kim, H. (2023). Few-shot is enough: exploring ChatGPT prompt engineering method for automatic question generation in english education. *Education and Information Technologies*, 1–33. https://doi.org/10.1007/s10639-023-12249-8

Lei, H., Cui, Y., & Zhou, W. (2018). Relationships between student engagement and academic achievement: A meta-analysis. *Social Behavior and Personality*, *46*(3), 517–528. https://doi.org/10.2224/sbp.7054

Liu, J., Xia, C. S., Wang, Y., & Zhang, L. (2023). *Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation*. http://arxiv.org/abs/2305.01210

Lu, L., Neale, N., Line, N. D., & Bonn, M. (2022). Improving Data Quality Using Amazon Mechanical Turk Through Platform Setup. *Cornell Hospitality Quarterly*, *63*(2), 231–246. https://doi.org/10.1177/19389655211025475

Mcshane, L., & Lopez, C. (2023). Perceived complexity of 3d shapes for spatial visualization tasks: humans vs generative models. *Proceedings of the ASME IDETC-CIE 2023*, 1–10.

Oliveira, W., Hamari, J., Shi, L., Toda, A. M., Rodrigues, L., Palomino, P. T., & Isotani, S. (2023). Tailored gamification in education: A literature review and future agenda. *Education and Information Technologies*, *28*(1), 373–406. https://doi.org/10.1007/s10639-022-11122-4

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., … Zoph, B. (2023). *GPT-4 Technical Report*. http://arxiv.org/abs/2303.08774

Ortolan, P. (2023). *Optimizing Prompt Engineering for Improved Generative AI Content*. [Trabajo de fin de grado, Universidad Pontificia Comillas]. http://hdl.handle.net/11531/80629

Saleem, A. N., Noori, N. M., & Ozdamli, F. (2022). Gamification Applications in E-learning: A Literature Review. *Technology, Knowledge and Learning*, *27*(1), 139–159. https://doi.org/10.1007/s10758-020-09487-x

Sarsa, S., Denny, P., Hellas, A., & Leinonen, J. (2022). Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. *ICER 2022 - Proceedings of the 2022 ACM Conference on International Computing Education Research*, 1, 27–43. https://doi.org/10.1145/3501385.3543957

Scherer, R., Siddiq, F., & Sánchez-Scherer, B. (2021). Some Evidence on the Cognitive Benefits of Learning to Code. *Frontiers in Psychology*, 12. https://doi.org/10.3389/fpsyg.2021.559424

Shankar, S., Zamfirescu-Pereira, J. D., Hartmann, B., Parameswaran, A. G., & Arawjo, I. (2024). *Who Validates the Validators? Aligning LLM-Assisted Evaluation of LLM Outputs with Human Preferences*. http://arxiv.org/abs/2404.12272

Shieh J. (2023). *Best practices for prompt engineering with the OpenAI API | OpenAI Help Center*. OpenAI. https://bit.ly/4cSZyg6

Shin, E., & Ramanathan, M. (2023). Evaluation of prompt engineering strategies for pharmacokinetic data analysis with the ChatGPT large language model. *Journal of Pharmacokinetics and Pharmacodynamics*, 51. https://doi.org/10.1007/s10928-023-09892-6

Sinclair, J., Butler, M., Morgan, M., & Kalvala, S. (2015). Student Engagement in computer science. *Annual Conference on Innovation and Technology in Computer Science Education, ITiCSE*, *2015-June*, 242–247. https://doi.org/10.1145/2729094.2742586

Sweller, J. (1988). Cognitive Load During Problem Solving: Effects on Learning. *Cognitive Science*, *12*(2), 257–285. https://doi.org/10.1207/s15516709cog1202_4

Velasquez-Hainao, J. D., Franco-Cardona, C. J., & Cadavid-Higuita, L. (2023). Prompt Engineering: a methodology for optimizing interactions with AI-Language Models in the field of engineering. *DYNA*, 1–9.

Wang, S., Xu, T., Li, H., Zhang, C., Liang, J., Tang, J., Yu, P. S., & Wen, Q. (2024). *Large Language Models for Education: A Survey and Outlook*. http://arxiv.org/abs/2403.18105

Yazidi, A., Abolpour Mofrad, A., Goodwin, M., Hammer, H. L., & Arntzen, E. (2020). Balanced difficulty task finder: an adaptive recommendation method for learning tasks based on the concept of state of flow. *Cognitive Neurodynamics*, *14*(5), 675–687. https://doi.org/10.1007/s11571-020-09624-3

Zhan, Z., He, L., Tong, Y., Liang, X., Guo, S., & Lan, X. (2022). The effectiveness of gamification in programming education: Evidence from a meta-analysis. In *Computers and Education: Artificial Intelligence* (Vol. 3). Elsevier B.V. https://doi.org/10.1016/j.caeai.2022.100096

Zhang, R., Guo, J., Chen, L., Fan, Y., & Cheng, X. (2022). A Review on Question Generation from Natural Language Text. *ACM Transactions on Information Systems*, *40*(1). https://doi.org/10.1145/3468889

Zhou, Y., Muresanu, A. I., Han, Z., Paster, K., Pitis, S., Chan, H., & Ba, J. (2022). *Large Language Models Are Human-Level Prompt Engineers*. http://arxiv.org/abs/2211.01910

Zu, T., Munsell, J., & Rebello, N. S. (2021). Subjective Measure of Cognitive Load Depends on Participants' Content Knowledge Level. *Frontiers in Education*, 6, 647097. https://doi.org/10.3389/feduc.2021.647097

AUTHORS' CONTRIBUTIONS, FINANCING AND ACKNOWLEDGMENTS

Author's contributions:

Conceptualization: Lopez, Christian y Morrison, Miles. **Validation**: Lopez, Christian. **Formal Analysis:** Lopez, Christian. **Data Curation:** Lopez, Christian; Morrison, Miles y Deacon, Matthew**; Preparation of original draft:** Morrison, Miles y Deacon, Matthew **Redaction and editing:** Lopez, Christian y Morrison, Miles; **Visualization:** Lopez, Christian. **Supervision:** Lopez, Christian **Project Administration:** Lopez, Christian. **All the authors have read and accepted the publish version of the manuscript:** Lopez, Christian; Morrison, Miles y Deacon, Matthew.

Financing: This study was supported by external grant from the Dominican Republic (FONDOCYT 2022-3A1-112).

Acknowledgements: This project is supported by the National Fund for Innovation and Scientific and Technological Development (FONDOCyT for its acronym in Spanish) from the Ministry of Higher Education, Science, and Technology of the Dominican Republic (project FONDOCYT 2022-3A1-112: "Machine Learning and Gamification for a personalized adaptive educational application"), and is in collaboration with the Universidad Nacional Pedro Henríquez Ureña (UNPHU) from the Dominican Republic.

AUTOR/ES:

Christian Lopez
Lafayette College.
Universidad Nacional Pedro Henríquez Ureña (UNPHU).

He is an Assistant Professor of Computer Science with an affiliation in Mechanical Engineering at Lafayette College. His research interests are in the design and optimization of intelligent decision support systems and persuasive technologies to augment human proficiencies. What this means is, he works on designing and creating systems to help make better decisions and help improve task performance by integrating technologies and methods from science and engineering, such as Machine Learning and Virtual Reality. In some cases, these systems need to be able to motivate individuals as well; hence, the use of persuasive technologies like gamification.
lopezbec@lafayette.edu

Índice H: 13
Orcid ID: https://orcid.org/0000-0003-2801-4618
Google Scholar: https://scholar.google.com/citations?user=t2ZEe1MAAAAJ&hl=en&oi=sra
ResearchGate: https://www.researchgate.net/profile/Christian-Lopez-B

Miles Morrison
Lafayette College.

Miles Morrison is pursuing an undergraduate degree in Integrative Engineering with a Robotics Focus at Lafayette College in Easton, PA, and is expected to graduate in 2026. He intends to pursue a graduate degree after obtaining his bachelor's from Lafayette College to further his expertise. This is his first official contribution to research work and will likely contribute to more in the future. His research and professional interests include applications of artificial intelligence, robotics; digital automation, and systems optimization.
morrismp@lafayette.edu

Matthew Deacon
Lafayette College.

Matthew Deacon is pursuing an undergraduate degree in Mechanical Engineering with a minor in Economics at Lafayette College in Easton, PA, and is expected to graduate in 2026. He intends to pursue an MBA after obtaining his bachelor's degree. In the summer of 2021, Matthew completed a paper on Stroke data for Prof. Guillermo Goldsztein from Georgia Tech as part of the Data Science and Machine Learning Course for Horizon Inspires Academic. He also completed an online course called "Programming for Everybody - Getting started with Python" through the University of Michigan. Matthew's professional interests include the use of engineering to innovate and create new products, applications or technologies.
deaconmp@lafayette.edu