Geração de questões de programação baseada em templates e IA generativa

Abner Santana[1], Francisco Genivan Silva[12], Jadson Lucas Gomes Souza[1], Júlio César da S. Dantas[1], Eduardo Henrique da Silva Aranha[1]

[1]Universidade Federal do Rio Grande do Norte
Departamento de Informática e Matemática Aplicada

[2]Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Norte

abnersant10@gmail.com, genivan.silva@ifrn.edu.br, jadson.souza@ufrn.br

dantas.j.c.s@gmail.com, eduardoaranha@dimap.ufrn.br

Abstract. *Teaching programming is a complex task due to the very nature of the subject. There is a constant need for students to practice through questions in order to absorb the content, creating a high demand on teachers. This article aims to automate this process using Large Language Models (LLMs), enhancing learning in a personalized manner. Foundational studies on automatic question generation were based on rule-based systems and pattern matching; recent advances employ natural language processing and deep learning techniques to produce more diverse and context-aware questions, with LLMs becoming a widely adopted and relevant approach for this field. The system seeks to provide coherent and contextualized questions based on specific topics, respecting pedagogical constraints that ensure the gradual development of the concepts covered. Questions were generated following a template-based model, allowing greater control over both the quality and variety of the exercises. Prompt Engineering techniques were used to ensure that each question followed a fixed format (topic, difficulty, context, problem, and variables). Among the main issues identified, the following stand out: (i) the generation of incomplete or ambiguous questions; (ii) the premature introduction of advanced concepts in more basic topics, which goes against the expected learning progression; and (iii) the model's difficulty in maintaining a diversity of problems within the same context without overly similar reformulations. Experts evaluated a sample of questions based on factors such as relevance to learning objectives, clarity, and level of difficulty, deeming the model appropriate for teaching programming.*

Resumo. *Ensinar programação é uma tarefa complexa devido à própria natureza do assunto. Há uma necessidade constante de que os alunos pratiquem por meio de questões para absorver o conteúdo, criando uma alta demanda sobre os professores. Este artigo visa automatizar esse processo utilizando Modelos de Linguagem de Grande Escala (LLMs), aprimorando o aprendizado de maneira personalizada. Estudos fundamentais sobre geração automática de questões foram baseados em sistemas baseados em regras e correspondência de padrões; avanços recentes empregam técnicas de processamento de linguagem natural e aprendizado profundo para produzir questões mais diversificadas e sensíveis ao contexto, sendo que os LLMs tornaram-se uma abordagem*

DOI: 10.5753/wie.2025.13388

*amplamente adotada e relevante para este campo. O sistema busca fornecer questões coerentes e contextualizadas com base em tópicos específicos, respeitando restrições pedagógicas que garantem o desenvolvimento gradual dos conceitos abordados. As questões foram geradas seguindo um modelo baseado em templates, permitindo maior controle tanto sobre a qualidade quanto sobre a variedade dos exercícios. Técnicas de *Prompt Engineering* foram utilizadas para garantir que cada questão seguisse um formato fixo (tópico, dificuldade, contexto, problema e variáveis). Entre os principais problemas identificados, destacam-se: (i) a geração de questões incompletas ou ambíguas; (ii) a introdução prematura de conceitos avançados em tópicos mais básicos, o que vai contra a progressão de aprendizagem esperada; e (iii) a dificuldade do modelo em manter uma diversidade de problemas dentro do mesmo contexto sem reformulações excessivamente semelhantes. Especialistas avaliaram uma amostra de questões com base em fatores como relevância para os objetivos de aprendizagem, clareza e nível de dificuldade, considerando o modelo apropriado para o ensino de programação.*

1. Introdução

É sabido que os exercícios desempenham um papel importante na aprendizagem; de acordo com [Thalheimer 2003], eles ajudam os estudantes a focar a atenção no material de estudo, oferecem prática na recuperação da informação relevante da memória, fornecem *feedback* para corrigir eventuais equívocos e proporcionam aos aprendizes informações sobre sua capacidade de recuperar conteúdos, o que pode motivá-los a se engajar em atividades adicionais de aprendizagem. No entanto, a geração manual de exercícios é uma tarefa que exige experiência, recursos e uma quantidade considerável de tempo [Kurdi et al. 2020].

Diante dessas limitações, pesquisadores têm explorado abordagens automatizadas para agilizar o processo de geração de perguntas. Embora a resposta automática a perguntas seja amplamente estudada na comunidade de processamento de linguagem natural, a geração de perguntas permanece menos explorada. Apesar de seu *status* de nicho, ela possui aplicações valiosas, desde a melhoria de sistemas de QA até o apoio à criação de conteúdos educacionais; automatizar esse processo pode reduzir a carga de trabalho dos educadores e aprimorar a aprendizagem [Kurdi et al. 2020].

Enquanto os sistemas tradicionais de geração automática de perguntas são baseados em regras, sintaxe, semântica, templates e modelos estatísticos mais simples, pesquisas recentes migraram para o uso de grandes modelos de linguagem. Apesar das muitas vantagens destacadas na literatura, essas aplicações ainda enfrentam desafios comuns associados aos LLMs, como a falta de uniformidade nas respostas e as alucinações. Em [Meißner et al. 2024], por exemplo, foram necessários diversos *prompts* e uma abordagem manual e iterativa para realizar a tarefa, e mesmo assim os autores relatam que alguns modelos falham em seguir padrões de design de programação orientada a objetos, em corresponder ao nível de dificuldade indicado no *prompt* e em identificar e classificar erros no código. Embora o uso de *templates* seja um método amplamente utilizado devido à sua natureza estruturada e versátil, eles apresentam a desvantagem de exigir uma quantidade substancial de trabalho humano especializado; projetar *templates* eficazes demanda

tempo e expertise para garantir que sejam adaptáveis a diferentes contextos, mantendo clareza e relevância.

Para superar esses desafios, a abordagem aqui descrita combina a estrutura de uma técnica baseada em templates com grandes modelos de linguagem, usados para preencher esses templates. Ela também incorpora outros aspectos menos avaliados na literatura, como o uso de *threads* para promover diversidade nas perguntas e a divisão da geração em dois *prompts* distintos: o primeiro cria uma variedade de contextos para as perguntas e o segundo expande esses contextos, elaborando a pergunta propriamente dita. O *framework* é projetado de forma a aproveitar o método rígido e bem estruturado dos templates, enfrentando sua dependência de conteúdo criado por humanos com o uso de grandes modelos de linguagem, que são, por sua vez, contidos pelos próprios *templates*, minimizando assim as inconsistências nas respostas.

Dito isso, esta pesquisa tem como objetivo investigar a capacidade de uma aplicação de IA Generativa na geração de questões de programação, avaliando as perguntas geradas pelo método proposto em termos de clareza, originalidade, foco temático, dificuldade e relevância. A avaliação foi realizada por quatro professores vinculados à um programa de pós-graduação.

2. Trabalhos relacionados

Este artigo expande o trabalho anterior de [Santana et al. 2025], contribuindo com uma avaliação mais abrangente das questões geradas. No estudo original, foi relatada uma experiência iterativa de aplicação de técnicas de *prompt* para a geração de questões de programação. Neste trabalho, parte-se de uma versão adaptada da escolhida naquele trabalho para construir um banco de questões, que é submetido à avaliação de especialistas. Em outras palavras, enquanto o foco anterior estava no processo de refinamento e avaliação dos *prompts*, o foco agora recai sobre a avaliação quantitativa das questões geradas — uma etapa não abordada anteriormente e que constitui a principal contribuição deste artigo.

2.1. Geração automática de questões

A geração automática de questões (GAQ) busca automatizar a criação de grandes volumes de questões de qualidade, empregando algoritmos que exploram fontes de conhecimento estruturadas ou não estruturadas [Kurdi et al. 2020]. Suas técnicas principais incluem abordagens sintáticas, semânticas, baseadas em regras, estatísticas e em templates. As abordagens sintáticas utilizam informações gramaticais para gerar questões plausíveis, ainda que com coerência semântica limitada [Huang and Mostow 2015, Mostow et al. 2017, Kurdi et al. 2020], enquanto as semânticas recorrem ao significado e a ontologias para criar distratores relacionados [Kurdi et al. 2020], como no sistema híbrido de [Odilinye et al. 2015]. Métodos baseados em regras transformam sentenças conforme operações linguísticas predefinidas, assegurando correção e pertinência [Chinkina et al. 2017], e os estatísticos aprendem padrões de geração a partir de dados, combinando similaridades semânticas e sintáticas para produzir alternativas coerentes [Kumar et al. 2015, Naeiji et al. 2023].

2.2. Geração de questões baseada em templates

A geração baseada em *templates* utiliza estruturas frasais fixas com lacunas preenchidas conforme características sintáticas ou semânticas [Kurdi et al. 2020]. [Ai et al. 2015], por exemplo, desenvolveu *templates* derivados de padrões de extração de relações, aplicando técnicas de parafraseamento para diversificar as questões mantendo a relação semântica subjacente. De modo semelhante, [Fatih and Romadhony 2023] criou um gerador automático de questões de verdadeiro/falso em indonésio, preenchendo espaços reservados com informações-chave do texto e utilizando técnicas como substituição por sinônimos, negações ou modificações numéricas, para avaliar diferentes aspectos da compreensão.

2.3. Geração automática de questões com modelos de linguagem de grande escala (*Large Language Models*, LLMs)

Com a popularização de modelos de linguagem de grande escala (LLMs), novas abordagens para GAQ surgiram. [Chan et al. 2023] avaliou o uso do ChatGPT para gerar questões abertas, analisando critérios como relevância, concisão e completude, além de propor um avaliador automatizado baseado em GPT para classificar questões, reduzindo custos e esforço humano. De forma semelhante, [Meißner et al. 2024] comparou o desempenho de *ChatGPT*, *Bing AI Chat* e *Google Bard* na criação de exercícios de programação, destacando o potencial dos LLMs na agilização do processo, embora reforçando a necessidade de supervisão humana.

[Luen William and Lim 2024] analisou *Mixtral* e *LLaMa2* na geração de questões de programação, considerando aspectos como formatação, correção e clareza. Em múltipla escolha, [Niu and Xue 2023] utilizou o *ChatGPT* aliado ao *Rasch model* para criar exercícios personalizados de sistemas operacionais, adequando a dificuldade ao nível do estudante. Por fim, [Doughty et al. 2024] investigou a geração automática de questões de programação em *Python*, usando critérios como clareza e qualidade dos distratores, e constatou que o *GPT-4* é capaz de produzir questões bem estruturadas e alinhadas aos objetivos de aprendizagem.

3. Metodologia

A metodologia para o desenvolvimento e avaliação da ferramenta automatizada de geração de questões de programação, baseada em modelos de linguagem de grande escala, se baseia no trabalho de [Santana et al. 2025] e foi estruturada em três etapas interconectadas: (1) criação e refinamento iterativos do prompt, (2) implementação técnica da ferramenta utilizando a *API* da *OpenAI*, e (3) validação especializada da qualidade das questões. Cada fase foi concebida para assegurar que as questões geradas fossem pedagogicamente alinhadas, desprovidas de ambiguidades e adaptáveis a diferentes níveis de complexidade, mantendo a coerência com tópicos específicos de programação.

3.1. Criação e Refinamento do *Prompt*

O *prompt* foi desenvolvido por meio de ciclos iterativos, com foco na clareza instrucional e na redução de ambiguidades. Diretrizes rigorosas, validadas por meio de testes internos, foram estabelecidas para orientar o modelo na geração de contextos realistas, problemas estruturados e variáveis adaptáveis. Os principais elementos incluem:

Geração de Contexto: Cenários que simulam aplicações práticas, como otimização de sistemas ou desafios do cotidiano.

Extração de Problemas: Definição de tarefas derivadas do contexto, assegurando a relevância temática.

Identificação de Variáveis: Listagem de componentes dinâmicos e personalizáveis que permitem testar múltiplos casos (por exemplo, $x = 5$, $y = -3$) e gerar variações da questão, preservando a estrutura lógica original.

Coerência Pedagógica: Restrições para evitar incongruências (por exemplo, questões condicionais não devem exigir o uso de laços).

O *prompt* final incorpora um modelo padronizado, ilustrado parcialmente abaixo. Este trecho destaca a estrutura central, enquanto o *prompt* completo inclui instruções detalhadas (por exemplo, regras para evitar ambiguidades, definições hierárquicas de temas, exemplos de problemas válidos e inválidos) que orientam o modelo durante a execução:

> **Você é um especialista em educação em programação e criará um banco de questões estruturadas com base em templates.**
> Cada questão gerada deve seguir **estritamente** esta estrutura:
> **Tópico:** {tópico}
> **Título:** [Curto e conciso]
> **Dificuldade:** {dificuldade}
> **Contexto:** [Um cenário realista onde múltiplos problemas podem surgir]
> **Problema:** [Defina um problema específico derivado do contexto, solucionável por meio de programação]
> **Variáveis:** [Liste valores modificáveis para adaptar o problema]

O *prompt* atua como o núcleo pedagógico e técnico da ferramenta, assegurando que as questões geradas alcancem um equilíbrio entre a flexibilidade criativa e a conformidade com os padrões educacionais. Sua estrutura foi validada não apenas por meio de testes iterativos, mas também através da implementação prática no código, onde regras claras (por exemplo, extração de variáveis via expressões regulares) e restrições temáticas (proibição de laços em questões condicionais) garantem que cada problema gerado funcione como um recurso educacional coeso e escalável. Essa abordagem possibilitou a tradução de diretrizes abstratas em resultados concretos, alinhando as capacidades generativas do modelo de linguagem com objetivos educacionais predefinidos.

3.2. Geração de Questões

Para gerar um banco estruturado de questões utilizando templates, desenvolvemos uma ferramenta baseada em Python integrada à *API* da *OpenAI* (modelo *GPT-4*). Esta ferramenta otimizou todo o processo, permitindo testes eficientes do *prompt*, padronização das saídas, comunicação controlada com o modelo e uso otimizado de tokens. Diversas funcionalidades técnicas foram incorporadas para garantir confiabilidade e escalabilidade, incluindo mecanismos de gerenciamento de mensagens e geração de questões orientada por templates.

O processo de geração de questões opera por meio de uma estrutura unificada que combina regras dos *templates*, protocolos de comunicação e integração com a *API*. Para fins de clareza, segmentamos a metodologia em dois componentes: a arquitetura técnica para o gerenciamento de mensagens com a *API* da *OpenAI* e o procedimento para

a geração das questões. Essa divisão destaca os princípios de design do sistema, com foco na eficiência do backend e no processamento em lote para assegurar a escalabilidade.

Inicialmente, o assistente é configurado com um *prompt* detalhado. Uma *thread* isolada é então criada para cada interação, com mensagens enviadas e processadas com base no histórico da *thread* para gerar uma resposta. O fluxo incorpora mecanismos de *timeout* para lidar com instabilidades da API, garantindo que o processo seja reiniciado em caso de falha.

O fluxo operacional completo inicia-se com o assistente gerando um contexto realista, extenso e rico em informações, que serve como base para múltiplos problemas potenciais, sem definir previamente um desafio específico. A partir desse contexto, são extraídos problemas e variáveis para assegurar rigorosa aderência ao tema e ao nível de dificuldade especificados. Questões validadas são armazenadas em um formato estruturado (por exemplo, uma planilha), enquanto respostas incompletas ou não conformes acionam um processo de regeneração.

Essa abordagem nos permitiu gerar, de forma eficiente, questões que seguiram o conceito baseado em templates, garantindo consistência e adaptabilidade em diversos tópicos e níveis de dificuldade em programação. Ao aproveitar as capacidades da ferramenta, pudemos focar no refinamento da qualidade e do valor pedagógico das questões, em vez de nos concentrarmos nas complexidades técnicas do processo de geração.

3.3. Validação por Especialistas

Para avaliar a qualidade das questões de programação geradas, foram criadas 60 questões - divididas entre os temas de operadores e operações matemáticas, condicionais, repetidores, listas e funções, sendo distribuídas equitativamente entre quatro especialistas. Todos os especialistas são vinculados ao programa de Pós-graduação em Sistemas e Computação da UFRN, com experiências diversas que perpassam o ensino básico, técnico e superior em computação. Os critérios de avaliação, pontuados com uma rúbrica em escala de Likert (1 a 4), incluíram:

Clareza: Ausência de ambiguidades na formulação.

Foco Temático: Restrição aos conceitos de programação especificados.

Originalidade: Originalidade na aplicação dos conceitos.

Relevância Prática: Conexão com cenários do mundo real.

Dificuldade: Exigência de raciocínio além da mera aplicação sintática.

Os resultados dessa avaliação serão utilizados para refinar os parâmetros do *prompt* e calibrar a ferramenta, garantindo a eficácia pedagógica.

4. Resultados

Os resultados a seguir são analisados segundo os temas abordados nas questões: operadores e operações matemáticas, condicionais, repetição, listas e funções.

4.1. Operadores e operações matemáticas

As questões que discutiram o uso de operadores e operações matemáticas foram as melhores avaliadas no quesito de clareza, com 92% das questões alcançando o conceito

máximo: O problema é descrito de forma concisa, com termos técnicos precisos, exemplos (se necessários) e estrutura lógica. Todas as regras, entradas, saídas e restrições são explicitadas sem redundâncias ou contradições. Não há espaço para interpretações alternativas.

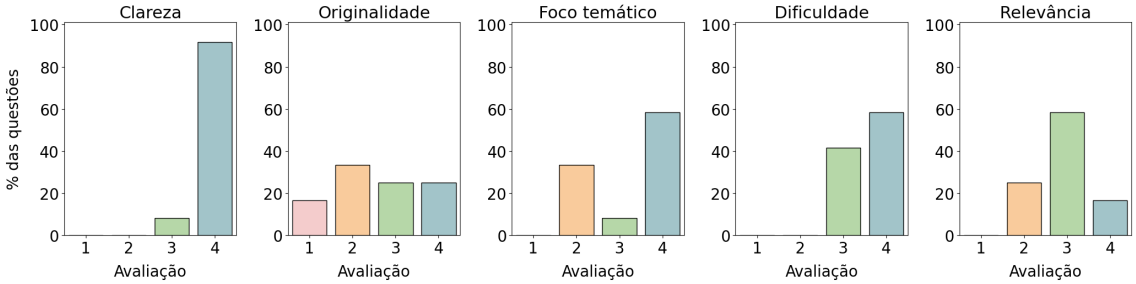

Figure 1. Resultados das avaliações de questões sobre operadores.

Essas questões, no entanto, apresentam o maior valor de desvio padrão para o quesito originalidade, e menor média, com valor de 2.5. Os resultados nos quesitos originalidade e clareza podem estar associados com a simplicidade que essas questões exigem por discutirem conhecimentos basilares.

No quesito dificuldade, todas as avaliações são positivas, sugerindo a capacidade da abordagem de se ater à restrições de complexidade para as questões nesse tema. Nos quesitos Foco temático e Relevância, tem desempenho menos uniforme mas positivo, com médias de 3.25 e 2.92, respectivamente.

4.2. Estruturas condicionais

De modo geral, as questões sobre estruturas condicionais foram muito bem avaliadas. Nenhum dos quesitos apresentou a avaliação mais baixa, e apenas duas, dentre as 12 questões avaliadas, receberam conceito 2.

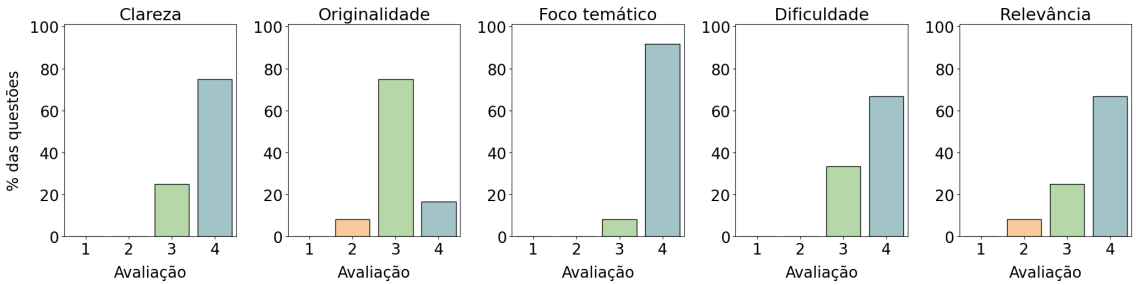

Figure 2. Resultados das avaliações de questões sobre estruturas condicionais.

As questões deste grupo apresentam a segunda maior média no quesito clareza e empatam com o grupo de questões sobre listas no melhor desempenho em todos os demais quesitos. Um comentário recorrente dos avaliadores destacou o desalinhamento entre o nível de dificuldade das questões e o nível pretendido; esse aspecto talvez sugira a necessidade de adoção de outros mecanismos de controle da dificuldade mais eficientes.

4.3. Estruturas de repetição

O grupo de questões sobre estruturas de repetição apresentou o pior desempenho nos quesitos Foco temático e Dificuldade, com médias de 2.83 e 2.92, respectivamente. Considerando os desvios-padrão de 0.94 e 0.92, conclui-se que a percepção de uma boa avaliação desse grupo esbarra em muita variabilidade.

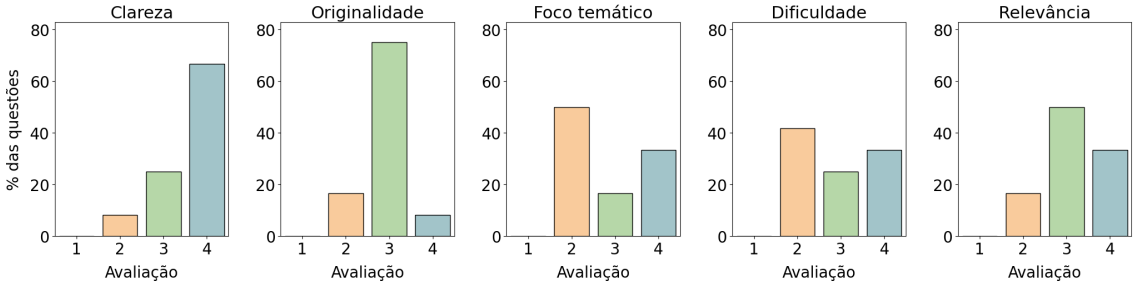

Figure 3. Resultados das avaliações de questões sobre estruturas de repetição.

Mais especificamente no quesito de Foco temático, o erro mais comum foi a necessidade do uso de estruturas ainda não conhecidas pelos estudantes; metade das questões avaliadas recebeu o conceito 2: inclui tópicos não ensinados até o momento ou adições irrelevantes que desviam do tema principal. Nas palavras dos avaliadores:

> **Avaliador 1:** A questão pede que seja exibida uma lista, objeto que ainda não foi aprendido no periodo de estruturas de repetição. Os alunos poderiam usar uma string para armazenar cada nova entrada de um novo usuário, mas essa abordagem não seria trivial de implementação se forem consideradas multiplas entradas para o mesmo usuário. Considerando que a questão é de nível Fácil, sou levado a crer que houve um erro nas estruturas usadas pela resposta da questão.
> **Avaliador 2:** A resposta deve incluir a leitura de uma lista, o que não está alinhado com o foco temático.

Essa é uma limitação importante no contexto pedagógico, frequentemente apontada por outros autores. Superá-la é fundamental para viabilizar o uso da IA generativa como um motor para a criação automática de questões. Afinal, se os professores não puderem ter confiança de que as questões geradas estão alinhadas ao nível de instrução dos estudantes e ao currículo proposto, essa solução não poderá ser considerada plenamente eficaz em seus objetivos.

4.4. Listas

Como dito acima, o grupo de questões sobre listas empata com o de estruturas condicionais com as melhores avaliações em Originalidade, Foco temático, Dificuldade e Relevância, embora apresente maior média de desvio-padrão.

A qualidade dos grupos anteriores no quesito Foco temático foi por vezes diminuída pela exigência de listas. Por isso, para o grupo de questões desse tema, é de se esperar que o resultado no quesito fosse razoável, com nenhuma avaliação negativa. Um problema comum identificado pelos avaliadores foi a falta de clareza sobre a entrada de

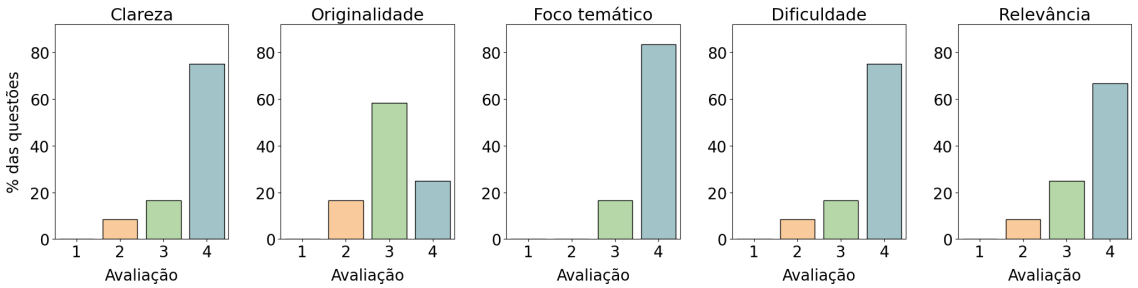

Figure 4. Resultados das avaliações de questões sobre listas.

dados exigida pela questão: se deveriam existir multiplos inputs ou um único, passando a lista como *string*. Nas palavras dos avaliadores:

> **Avaliador 2:** A questão poderia ser um pouco mais clara sobre o formato das entradas.
> **Avaliador 4:** A questão não deixa claro se as quantidades serão passadas em outra lista, na mesma lista como um elemento conseguinte ao ingrediente ou, de maneira mais comum, como um dicionário, mas que não teria sido visto pelos estudantes.

4.5. Funções

As questões desse gurpo possuem boas médias para Foco temático e Relevância, não apresentando nenhuma avaliação negativa, com médias de 3.83 e 3.43, com Originalidade e Dificuldade possuindo valores menos uniformes mas ainda positivos - 2.75 e 3.50.

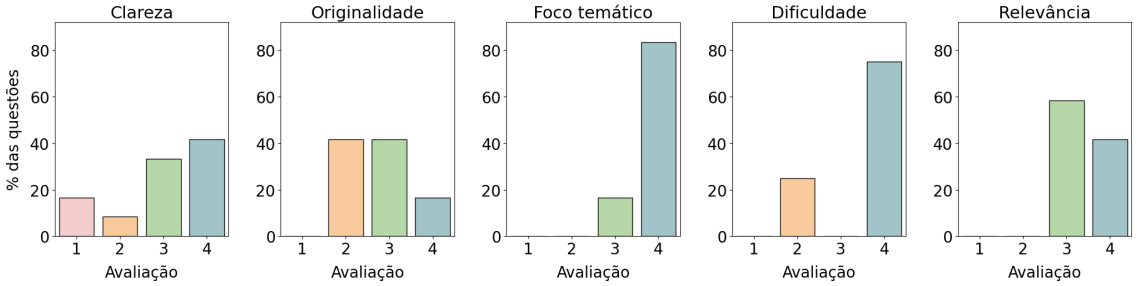

Figure 5. Resultados das avaliações de questões sobre listas.

No entanto, o grupo possui a maior divergência no quesito clareza, com desvio de 1.13 e com pontos críticos levantados pelos avaliadores:

> **Avaliador 1:** O enunciado não especifica como as funções deverão consumir a fonte da verdade, deixando a interpretação e tomada de decisão da solução para o leitor, haverá diversas soluções diferentes.
> **Avaliador 2:** Talvez fosse útil uma melhor definição dos tipos de dados que serão alimentados no sistema.
> **Avaliador 3:** Faltou fornecer uma informação importante para a resolução do problema.

5. Ameaças à validade

A principal limitação de validade interna do estudo decorre da subjetividade na avaliação qualitativa das questões geradas, potencialmente influenciada por vieses individuais e pela ausência de análise de confiabilidade entre avaliadores. A validade externa é restrita pelo escopo limitado a tópicos introdutórios de programação e pelo número reduzido de questões (60), o que compromete a generalização dos resultados para outros conteúdos ou contextos educacionais. Além disso, as avaliações foram conduzidas em ambiente controlado, sem aplicação prática, limitando conclusões sobre a efetividade pedagógica. Por fim, os achados refletem o desempenho de um modelo específico de IA generativa, cuja rápida evolução tecnológica pode tornar os resultados temporalmente desatualizados.

6. Considerações finais

As questões sobre operadores e operações matemáticas destacaram-se pela clareza, mas apresentaram limitações quanto à originalidade, possivelmente em decorrência da natureza mais elementar do conteúdo. Já as questões sobre estruturas condicionais obtiveram desempenho consistentemente positivo em todos os quesitos, ainda que com relatos sobre o desalinhamento entre o nível de dificuldade e o nível pretendido, indicando a necessidade de mecanismos adicionais de calibração da complexidade nos prompts.

No caso das estruturas de repetição, identificou-se o pior desempenho, particularmente no foco temático, associado à recorrentes desvios relacionados à exigência de conceitos não ainda ensinados, como o uso de listas. Esse achado revela uma limitação crítica para o uso pedagógico da IA generativa, que precisa ser superada para garantir que os itens produzidos sejam adequados ao estágio de aprendizagem dos estudantes.

O grupo de listas apresentou avaliações majoritariamente positivas, com destaque para a adequação temática e relevância. Contudo, problemas relacionados à clareza das instruções de entrada foram detectados, sugerindo que maior rigor na definição dos parâmetros das questões é necessário. De modo semelhante, o grupo de funções demonstrou desempenho robusto em foco temático e relevância, mas revelou fragilidades importantes no quesito clareza, refletidas pela alta variabilidade nas avaliações e pela necessidade de especificações mais detalhadas nos enunciados.

Esses achados reforçam que, embora promissora, a utilização de IA generativa na criação automática de questões demanda cuidados metodológicos adicionais, especialmente no que tange ao alinhamento entre o nível de complexidade exigido e o perfil do público-alvo, bem como à precisão na formulação dos enunciados.

Como trabalhos futuros, recomenda-se o desenvolvimento de estratégias para aprimorar o controle da complexidade das questões geradas, também a incorporação de validação automatizada da clareza e da adequação temática dos itens. Além disso, seria pertinente realizar avaliações pelos estudantes das questões geradas e estudos longitudinais, comparando o desempenho de estudantes expostos a questões geradas automaticamente com aqueles que utilizam itens elaborados por especialistas, para avaliar de forma mais abrangente a eficácia pedagógica da abordagem proposta.

References

Ai, R., Krause, S., Kasper, W., Xu, F., and Uszkoreit, H. (2015). Semi-automatic generation of multiple-choice tests from mentions of semantic relations. In *2nd Workshop on*

Natural Language Processing Techniques for Educational Applications, pages 26–33.

Chan, W., An, A., and Davoudi, H. (2023). A case study on chatgpt question generation. In *2023 IEEE International Conference on Big Data (BigData)*, pages 1647–1656.

Chinkina, M., Ruiz, S., and Meurers, D. (2017). Automatically generating questions to support the acquisition of particle verbs: Evaluating via crowdsourcing. In *CALL in a Climate of Change: Adapting to Turbulent Global Conditions*, pages 73–78.

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., and Sakr, M. (2024). A comparative study of ai-generated (gpt-4) and human-crafted mcqs in programming education. In *26th Australasian Computing Education Conference (ACE '24)*, pages 114–123, New York, NY, USA. Association for Computing Machinery.

Fatih, M. Z. A. and Romadhony, A. (2023). Automatic true/false question generation using template-based framework. In *2023 International Conference on Data Science and Its Applications (ICoDSA)*, pages 403–407, Bandung, Indonesia.

Huang, Y. T. and Mostow, J. (2015). Evaluating human and automated generation of distractors for diagnostic multiple-choice cloze questions to assess children's reading comprehension. In Conati, C., Heffernan, N., Mitrovic, A., and Verdejo, M., editors, *Artificial Intelligence in Education*, pages 155–164. Springer International Publishing, Cham.

Kumar, G., Banchs, R., and D'Haro, L. F. (2015). Automatic fill-the-blank question generator for student self-assessment. In *IEEE Frontiers in Education Conference (FIE)*, pages 1–3.

Kurdi, G., Leo, J., and Parsia, B. e. a. (2020). A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204.

Luen William, C. W. and Lim, T. M. (2024). Comparative studies: Leveraging large language model in theoretical and practical assessment sample question-answer bank on programming related subjects. In *IEEE 4th International Conference on Electronic Communications, Internet of Things and Big Data (ICEIB)*, pages 331–335, Taipei, Taiwan.

Meißner, N., Speth, S., and Becker, S. (2024). Automated programming exercise generation in the era of large language models. In *36th International Conference on Software Engineering Education and Training (CSEET)*, pages 1–5, Würzburg, Germany.

Mostow, J., Huang, Y. T., Jang, H., Weinstein, A., Valeri, J., and Gates, D. (2017). Developing, evaluating, and refining an automatic generator of diagnostic multiple choice cloze questions to assess children's comprehension while reading. *Natural Language Engineering*, 23(2):245–294.

Naeiji, A., An, A., Davoudi, H., Delpisheh, M., and Alzghool, M. (2023). Question generation using sequence-to-sequence model with semantic role labels. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2830–2842, Dubrovnik, Croatia. Association for Computational Linguistics.

Niu, Y. and Xue, H. (2023). Exercise generation and student cognitive ability research based on chatgpt and rasch model. *IEEE Access*, 11:116695–116705.

Odilinye, L., Popowich, F., Zhang, E., Nesbit, J., and Winne, P. H. (2015). Aligning automatically generated questions to instructor goals and learner behaviour. In *IEEE 9th International Conference on Semantic Computing (ICS)*, pages 216–223.

Santana, A., Silva, F., Dantas, J., Souza, J., and Aranha, E. (2025). Geração automática de questões de programação usando llm: Um relato de experiência. In *Anais do XXXIII Workshop sobre Educação em Computação*, pages 1415–1425, Porto Alegre, RS, Brasil. SBC.

Thalheimer, W. (2003). The learning benefits of questions. Technical report, Work Learning Research.