Especialista em algoritmos para apoio interativo na aprendizagem de programação utilizando ChatGPT

Jean Miguel[1]**,Waldecir Martins**[1]**, Ícaro Benarrós**[1]**, José Carlos Duarte**[1,2]

[1]Universidade do Estado do Amazonas (UEA) - ThinkTEd Lab
Manaus – AM – Brasil

[2]Instituto de Computação – Universidade Federal do Amazonas (UFAM)
Manaus – AM – Brasil

{jmcb.lic22,wdsm.lic22,ibdo.lic23}@uea.edu.br

carlosduarte@icomp.ufam.edu.br

Abstract. *The increasing use of artificial intelligence in academia has generated discussions about its benefits and challenges as a learning tool. Given this, this study presents a programming expert designed to explain and guide problem-solving, avoiding the direct delivery of ready-made codes. Adopting an experimental approach, the study investigated how the expert can stimulate user autonomy and promote active learning. The results demonstrated that the expert prioritized conceptual explanations and logical flows, encouraging active learning by users. Although adjustments are needed to improve its efficiency, the results indicate that the expert has the potential to become an effective educational tool, acting as a promising facilitator in the programming learning process.*

Resumo. *O uso crescente de inteligência artificial no meio acadêmico tem gerado discussões sobre seus benefícios e desafios como ferramenta de aprendizagem. Diante disso, este estudo apresenta um especialista em programação, projetado para oferecer orientações na resolução de problemas, evitando a entrega direta de códigos. Adotando uma abordagem experimental, o estudo investigou como o especialista pode estimular a autonomia do usuário e promover uma aprendizagem ativa. Os resultados demonstraram que o modelo priorizou explicações conceituais e fluxos lógicos, incentivando o aprendizado ativo dos usuários. Embora sejam necessários ajustes para aprimorar sua eficiência, os resultados indicam que o especialista tem potencial para se tornar uma ferramenta educacional eficaz, atuando como um facilitador promissor no processo de aprendizagem em programação.*

1. Introdução

As interações sociais desempenham um papel essencial na construção do conhecimento, permitindo que os estudantes ampliem suas experiências e saberes por meio de relações com pessoas de diferentes perfis e níveis de conhecimento, como professores, colegas ou especialistas [Viecheneski and Carletto 2013]. Com o avanço das tecnologias, essas relações e interações estão sendo continuamente transformadas. Por isso, pesquisadores têm investigado como a inteligência artificial (IA) pode contribuir para a promoção de interações significativas entre o estudante e tecnologias como *chatbots*, criando melhores oportunidades para a construção do conhecimento [da Cruz et al. 2023]. Segundo

[Silveira et al. 2019], a IA pode atuar como recurso importante para motivar e incentivar estudantes na construção do conhecimento. Isso ocorre porque ela pode ser posicionada como um facilitador da aprendizagem, promovendo interações significativas entre o estudante e um agente conversacional [Delbone et al. 2024].

Dessas interações surgem benefícios que despertaram o interesse de estudantes e professores, não só pela facilidade de acesso [de Araujo 2024], mas também por vantagens como a personalização do ensino, a possibilidade de *feedback* imediato, a acessibilidade a conteúdos de qualidade e a melhoria do processo de aprendizagem [Picão et al. 2023]. No entanto, esse contato pode gerar preocupações como dependência dos estudantes na ferramenta, desestímulo de criatividade e perca do pensamento crítico [Lima 2023]. Para que a inteligência artificial seja posicionada de forma efetiva como facilitadora da aprendizagem, é necessário ressaltar como interações significativas são fundamentais nesse processo, pois um modelo generativo que entrega ao usuário todo tipo de informação sem incentivar o pensamento crítico e a resolução de problemas, não é capaz de atuar como facilitador de aprendizagem no contexto educacional.

Para enfrentar esse desafio, são necessárias estratégias educacionais complementares e integradas, como a criação de *chatbots* que não apenas forneçam respostas prontas, mas estimulem o estudante a refletir de forma autônoma e resolver problemas, servindo como um mediador e não substituta nesse processo, agindo como um guia enquanto deixa espaço para que o discente explore e descubra através de suas próprias ações. Além de fornecer respostas guiadas, com perguntas que incentivem o estudante a pensar mais profundamente sobre o problema, promovendo a autonomia e o pensamento crítico.

Nesse contexto, este estudo apresenta a seguinte questão de pesquisa: *Como o uso de um especialista em programação baseado em inteligência artificial, que não forneça respostas e códigos prontos, pode estimular a autonomia dos usuários e promover a aprendizagem ativa?*. Para responder, foi desenvolvido um agente conversacional que prioriza explicações conceituais e fluxos lógicos, em vez de fornecer respostas e algoritmos prontos, com o intuito de incentivar o aprendizado significativo e a resolução de problemas de forma independente. O estudo busca avaliar o potencial do especialista como uma ferramenta educacional promissora no ensino-aprendizagem de programação.

2. Trabalhos correlatos

A teoria socioconstrutivista de Vygotsky enfatiza que o aprendizado é um processo social, mediado por interações entre indivíduos e pela utilização de ferramentas culturais. Segundo a teoria, o conhecimento é construído de maneira colaborativa, com a mediação de um *"outro mais experiente"*, como professores, colegas ou até mesmo ferramentas tecnológicas, que auxiliam o estudante a alcançar sua zona de desenvolvimento proximal (ZDP) — o espaço entre o que ele já consegue realizar sozinho e o que pode realizar com apoio [Vigotsky et al. 1987].

Com o avanço das tecnologias digitais, essas ferramentas mediadoras têm se expandido, permitindo que o papel de mediador tradicional seja complementado por soluções baseadas em IA, como o ChatGPT, um modelo de linguagem desenvolvido pela OpenAI como agente conversacional do tipo generativa [Azaria 2022]. O modelo está cada vez mais presente no cotidiano escolar, servindo como monitor que ajuda a esclarecer dúvidas e facilitar o entendimento dos conteúdos [Ribeiro 2023]. Ao fornecer *fe-*

edback constante e desafios adequados ao nível de conhecimento do discente, a IA pode funcionar como um parceiro interativo que ajuda a potencializar a aprendizagem.

Nesse sentido, estudos estão sendo realizados para explorar a aplicação de IAs generativas na educação, visando estudar as interações com usuários, como no trabalho de [Lira et al. 2024], onde os autores investigaram o uso da ferramenta *GitHub Copilot*, como recurso de auxílio-aprendizagem em programação com estudantes de graduação do curso de ciência da computação. O objetivo foi avaliar a capacidade de resolução de problemas de programação, por meio de um experimento controlado onde os estudantes resolveram problemas com e sem o *GitHub Copilot*. Os resultados indicaram que ao utilizar a ferramenta, os estudantes, além de alcançarem mais acertos, também reduziram o tempo necessário para resolver os problemas. Esses achados mostraram que ferramentas baseadas em inteligência artificial podem auxiliar estudantes a resolver problemas; entretanto, surge a preocupação se essas soluções foram raciocinadas pelos próprios participantes ou desenvolvidas inteiramente pela ferramenta.

No trabalho de [Alves et al. 2024], houve o desenvolvimento de um especialista em gamificação para analisar *frameworks* de gamificação aplicados no contexto educacional. Para isso, utilizou de abordagens e prompts encontrados na literatura científica e a base de conhecimento do agente foi construída por artigos acadêmicos, o que indica um embasamento teórico sólido. Nesse perspectiva o agente apresentado neste estudo também seguiu essa metodologia, buscando estabelecer igualmente uma forte base teórica.

No entanto, se as ferramentas baseadas em IA continuarem sendo utilizadas para oferecer respostas automatizadas, isso pode ofuscar a relação mediador-estudante, não proporcionando desenvolvimento cognitivo significativo. A construção do conhecimento exige que esses modelos sejam projetados para promover o pensamento crítico, a criatividade e a autonomia, alinhando-se aos princípios socioconstrutivistas para promover uma aprendizagem ativa e reflexiva.

3. Metodologia

Este estudo seguiu uma abordagem experimental, visando explorar como um especialista em programação pode promover a autonomia do usuário e o aprendizado ativo, ajudando na solução de problemas sem entregar o código diretamente. A metodologia, apresentada na Figura 1 foi estruturada nas seguintes etapas: (i) desenvolvimento do especialista: foi criado um especialista no ChatGPT, projetado para priorizar respostas com explicações conceituais e fluxos lógicos, em vez de fornecer códigos prontos; (ii) aplicação do especialista: testes foram realizados com estudantes de ciência da computação, utilizando o agente como ferramenta de apoio para a resolução de problemas de programação; (iii) análise das interações: os dados coletados durante os testes foram analisados para entender o processo de interação entre os participantes e o especialista; e por fim, (iv) análise e descrição dos resultados: foram apresentados os resultados das análises realizadas, destacando como a ferramenta influenciou a autonomia dos participantes e se conseguiu promover um aprendizado ativo.

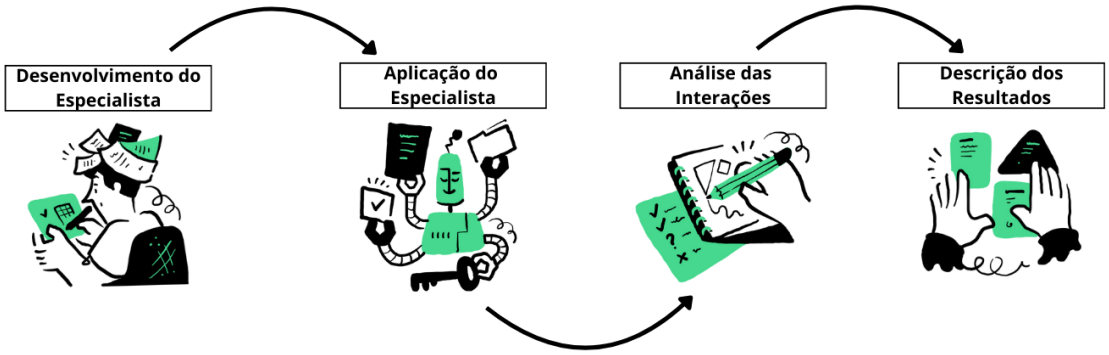


Figura 1. Processo Metódologico

3.1. Desenvolvimento do especialista virtual

O *Especialista*[1] foi desenvolvido utilizando o ChatGPT, que permite a criação de um *chatbot* customizado, ou seja, desenvolver uma ferramenta que pode ser personalizada com instruções que a guiarão durante seu funcionamento. Inicialmente, realizou-se um levantamento de estratégias de Engenharia de *prompt* para serem incorporadas ao modelo, foram usadas as abordagens descritas nos trabalhos de [Bassner et al. 2024] e [Liffiton et al. 2023] como principais referências de estratégias de Engenharia de *prompt*.

As estratégias utilizadas para moldar os comandos do especialista, foram: (i) *Few-shot Prompting* - técnica que consiste em fornecer alguns exemplos de entrada e saída para orientar o modelo em novas tarefas, permitindo que ele aprenda a lidar com elas com base nesses exemplos. Essa abordagem foi usada para apresentar exemplos conceituais e respostas contextualizadas, oferecendo orientações adicionais para tarefas que demandam maior suporte; e (ii) *Chain-of-Thought (CoT) Prompting* - permite que o modelo raciocine uma solução para um problema passo a passo, ajudando na criação de uma resposta coerente e promovendo um aprendizado significativo [Liang et al. 2023]. Essas abordagens foram combinadas para criar restrições que impedem o fornecimento de algoritmos ao usuário, priorizando explicações detalhadas que os orientem na construção de suas soluções e incentivar o pensamento crítico do usuário, visando promover interações significativas.

3.1.1. Validação do especialista

Foi realizado um estudo piloto com três usuários-testadores, que tinham o objetivo de interagir com o especialista para obter códigos prontos. A aplicação ocorreu por meio da resolução de exercícios de programação na linguagem *Python* utilizando o especialista como ferramenta de apoio; os participantes deveriam utilizar comandos que incentivassem o especialista a fornecer o código completo ou partes dele para a resolução dos problemas. Os exercícios podem ser consultados no material complementar [2].

Ao analisar as interações do grupo com o modelo, observou-se que ele apresentou algumas falhas durante as conversas, como fornecer trechos de código ou não explicar

[1]Acessar Especialista:https://chatgpt.com/g/g-mWTkPlqnW-especialista-em-algoritmos
[2]Material Complementar:https://doi.org/10.6084/m9.figshare.28464374.v2

claramente conceitos de programação. Todas as não conformidades identificadas foram corrigidas em novas versões do *prompt*. Em cada atualização do guia de instruções do especialista, novos testes eram realizados, visando atingir o resultado esperado. A evolução do *prompt* utilizado para compor o modelo desde sua concepção inicial até a versão final utilizada pode ser consultada no material complementar.

A versão do especialista atual realiza explicações conceituais para ajudar os estudantes e não fornece códigos prontos. Entretanto, alguns conceitos não são descritos de forma totalmente clara pelo modelo e, ocasionalmente, ele pode fornecer respostas prolixas ao incluir informações redundantes ou detalhes adicionais não solicitados pelo usuário. Porém, isso é comum em modelos projetados para dar respostas completas, como ao explicar conceitos ou fornecer contexto, como o especialista em questão.

3.2. *Design* Experimental: aplicação do especialista

Após o refinamento do *prompt* do modelo, foram realizados testes para avaliar o comportamento do especialista em suas interações com os usuários. A organização dos testes seguiu as etapas descritas abaixo:

i. Para este estudo forem selecionados três estudantes de graduação em computação, com diferentes níveis de conhecimento em programação. Os participantes foram alocados em uma sala equipada com um notebook contendo três guias abertas: a interface do especialista, uma aba do *Google Docs* com dois exercícios previamente definidos e a IDE *PyCharm*. No primeiro exercício, é necessário ler um valor de salário em ponto flutuante, calcular o novo salário, o valor do reajuste ganho e o percentual de reajuste aplicado conforme uma tabela de faixas salariais. A saída deve apresentar essas informações formatadas em três linhas, com duas casas decimais. No segundo exercício, deve-se ler um valor inteiro que representa a duração de um evento em segundos e convertê-lo para o formato horas:minutos:segundos. A saída deve ser exibida nesse formato, conforme os exemplos fornecidos
ii. Os estudantes receberam orientações para realizar as atividades utilizando exclusivamente o especialista como ferramenta de apoio. Eles foram informados de que poderiam elaborar *prompts* à sua escolha. Isso nos permitiu avaliar se o especialista respeitaria as restrições definidas em seu *prompt*.
iii. Os participantes tiveram liberdade para dedicar o tempo que considerassem necessário para completar as atividades ou optar por desistir sem finalizar os exercícios. Essa flexibilidade buscou refletir condições reais de uso e respeitar o ritmo individual de cada participante.
iv. Ao final do experimento, os participantes preencheram um formulário para avaliar sua experiência ao interagir com o especialista. O formulário foi elaborado para coletar dados qualitativos sobre a usabilidade, eficiência e potencial do especialista como ferramenta educacional, além de identificar possíveis melhorias.

3.2.1. Coleta e Análise das Interações

Os históricos das interações dos pariticpantes com o agente foram registrados e analisados, visando identificar padrões de comportamento e avaliar se as respostas fornecidas

pelo modelo contribuíram para o desenvolvimento da autonomia dos usuários. Além disso, a análise buscou verificar em que medida a ferramenta priorizou explicações conceituais e fluxos lógicos, alinhando-se ao propósito de incentivar o aprendizado significativo, sem fornecer respostas ou códigos prontos.

4. Resultados

A avaliação dos resultados foi realizada analisando o histórico de interações e as respostas do formulário utilizado para coletar a percepção dos usuários.

4.1. Análise das Interações

As interações foram analisadas em duas perspectivas: i) identificar como o especialista aplicou explicações conceituais e fluxos lógicos, sem fornecer algoritmos; ii) verificar se as respostas estimulavam a autonomia dos usuários, incentivavam os participantes a explorar conceitos de forma independente e guiavam o aprendizado de maneira significativa; e iii) as mesmas entradas que os participantes deram ao usar o especialista foram enviadas ao ChatGPT padrão (sem personalizações) para identificar as diferenças entre os comportamentos das ferramentas e comparar a qualidade das respostas.

A análise inicial das interações foi para descobrir com que frequência os participantes precisaram de esclarecimentos adicionais. A frequência foi avaliada observando os momentos em que os usuários retornaram com dúvidas ou pediram explicações adicionais após uma resposta inicial do agente.

APLICAÇÃO COM PARTICIPANTE 01
Número de interações: 11
Interações com dúvidas adicionais:

- Após entender o uso básico do input(), perguntou se poderia atribuir uma variável ao input().
- Questionou sobre o erro de TypeError.
- Perguntou como corrigir o erro.
- Questionou como exibir números formatados.
- Confirmou se a formatação poderia ser feita diretamente no print.

Frequência de dúvidas adicionais: 5 dúvidas em 11 interações (45%).

APLICAÇÃO COM PARTICIPANTE 02
Número de interações: 7
Interações com dúvidas adicionais:

- Perguntou sobre o funcionamento do código inicial (sal e rea).
- Questionou se o cálculo do aumento estava correto.
- Perguntou como formatar o valor exibido.

Frequência de dúvidas adicionais: 3 dúvidas em 7 interações (43%).

APLICAÇÃO COM PARTICIPANTE 03
Número de interações: 6
Interações com dúvidas adicionais:

- Perguntou sobre a organização da lógica das condições.

- Solicitou explicação sobre conversões de tempo.
- Perguntou como transformar segundos em HH:MM:SS.

Frequência de dúvidas adicionais: 3 dúvidas em 6 interações (50%).

Das 24 interações totais, 11 foram dúvidas adicionais, ou seja, uma frequência de 46%. Uma explicação para esses números pode ser atribuída ao fato de algumas respostas iniciais do *chatbot* podem não ter sido claras ou detalhadas o suficiente. Um exemplo a ser citado é a interação com o Participante 01, ele perguntou: *"Como eu leio uma entrada em Python?"*, o Chatbot respondeu: *"A leitura de dados em Python pode ser feita de várias maneiras, dependendo da origem dos dados: - Entrada de dados via teclado: receba informações digitadas diretamente pelo usuário. - Entrada de dados via arquivos: leia informações de arquivos externos. - Entrada de dados via parâmetros ou variáveis de ambiente: forneça informações ao programa no momento da execução [...]"*. Observando a interação, notou-se que a explicação não foi sucinta, necessitando de uma segunda entrada para entender a explicação.

A primeira interação do Participante 02 foi: *Estou fazendo a variável val receber um valor float. Como faço para printar esse valor?"*, já a primeira interação do Participante 03 foi *"Me ajude a fazer uma verificação em Python para essas condições"*, em ambas as situações a resposta do Chatbot começou com *"Sinto muito! Minhas diretrizes não permitem fornecer código em Python [...]"*, em seguida, ofereceu ajuda com *"Entretanto, posso orientá-lo sobre o conceito! [...]"* e *"No entanto, posso ajudá-lo a compreender a lógica por trás da verificação que você deseja realizar e guiá-lo para implementar isso por conta própria"[...]*. As interações seguintes dos participantes tornaram-se mais específicas, com perguntas como: *"O input() é utilizado em Python?"*. Alguns compartilharam trechos de seus códigos, e o chatbot respondeu, porém evitou fornecer exemplos práticos ou códigos completos, seguindo diretrizes que priorizam a explicação de conceitos em vez de respostas diretas.

Realizando uma análise das interações dos usuários com o modelo, pode-se destacar benefícios identificados nas respostas do especialista:

Orientação Conceitual: o chatbot ofereceu orientação conceitual durante as interações, onde as respostas seguiram uma estrutura didática, dividindo explicações em etapas claras. Esse tipo de explicação pode auxiliar o usuário a compreender o 'porquê' e o 'como', aspectos importantes para a construção de um conhecimento duradouro, além de facilitar a assimilação de conteúdos complexos em programação

Desenvolvimento da habilidade de compreender algoritmos: O *chatbot* demonstrou elementos que podem auxiliar no desenvolvimento da habilidade de compreender algoritmos, pois ao decorrer das interações, o modelo frequentemente abordou as soluções dos exercícios em etapas lógicas, o que pode ajudar o estudante a entender como abordar problemas de forma sistemática. Por exemplo, para o Participante 03, a explicação para transformar segundos em HH:MM:SS foi clara, dividindo os passos para calcular horas, minutos e segundos restantes:

Participante: *Como deve funcionar a lógica para transformar um número em segundos para o formato horas:minutos:segundos?*

Especialista: *Aqui está a lógica passo a passo para converter segundos em um*

formato horas:minutos:segundos:

- *Calcular as horas: cada hora tem 3600 segundos. Fórmula:* `Horas = Segundos // 3600`.
- *Calcular os minutos restantes: após calcular as horas, encontre os segundos restantes. Fórmula:* `Minutos = (Segundos % 3600) // 60`.
- *Calcular os segundos finais: Após calcular os minutos, o restante são os segundos finais:* `Segundos Finais = Segundos % 60`.
- *Formatar no formato HH:MM:SS: Combine as horas, minutos e segundos para exibir no formato desejado.*

Correção de códigos: Quando os usuários enviavam erros de compilação ao especialista, ele fornecia explicações detalhadas sobre as possíveis causas da falha, como erros de sintaxe ou problemas durante o processo de compilação. Por exemplo, Participante 01:

Participante: *Está me retornando o erro* `TypeError: can't multiply sequence by non-int of type 'float'`.

Especialista: *Esse erro ocorre porque o valor recebido de* `input()` *é uma string. Quando você tenta realizar uma operação matemática (como multiplicação), o Python não consegue processar diretamente a string como número, especialmente com números decimais (*`float`*).*

Por que o erro ocorre?

- `input()` *retorna uma string: Mesmo que o usuário digite um número, ele será tratado como texto.*
- *Operações inválidas com string e float: Quando você tenta multiplicar uma string por um número decimal (*`float`*), o Python não sabe como lidar com isso.*

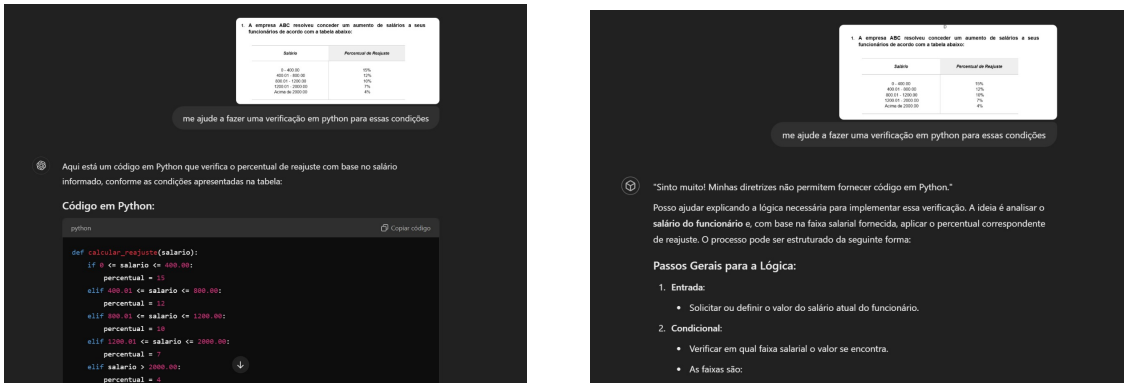
Como resolver?

- *Converter a entrada para número: se for um número inteiro, use* `int(a)`. *Se for um número decimal, use* `float(a)`.

Neste caso, o agente conversacional diagnosticou corretamente o problema e ofereceu uma solução prática, incentivando o usuário a entender a causa do erro antes de corrigi-lo, apesar de ocasionalmente apresentar um comportamento redundante nesse aspecto, pois frequentemente repete os mesmos motivos em situações similares, sem oferecer novos *insights*.

Construção da autonomia do usuário: O *chatbot* incentivou os usuários a pensar sobre a causa de erros, ao invés de oferecer respostas prontas. No exemplo acima, quando o usuário relatou o erro TypeError, o chatbot explicou que input() retorna uma string e sugeriu a conversão para float ou int, ajudando o usuário a entender por que o erro ocorreu. Porém, algumas explicações foram amplas demais, sem foco suficiente na dúvida do usuário, o que pode confundir ou desmotivar o usuário.

Comparações de respostas entre o especialista e o ChatGPT 4.0 foram realizadas para identificar se a ferramenta está cumprindo o objetivo desejado ou se o ChatGPT padrão seria suficiente. A Figura 2 ilustra uma dessas comparações, onde ambas as ferramentas receberam a mesma entrada.



(a) Interação com o ChatGPT 4 (b) Interação com o especialista

Figura 2. ChatGPT vs. especialista

Percebe-se ao analisar as imagens que o ChatGPT oferece o código completo após o usuário enviar o exercício e pedir auxílio em sua resolução. Contudo, ao especialista receber a mesma entrada, responde com instruções de como resolver o problema passo a passo, indo de explicações lógicas sobre o problema até dicas de como implementar o algoritmo.

4.2. Formulário de Percepção

O formulário de avaliação da ferramenta foi elaborado para coletar informações relevantes sobre o uso do especialista, onde para isso foi solicitada a autorização dos estudades para coleta e uso dos dadaos pessoais preenchidos durantes avaliação, dados esses que incluem informações sobre: (i) adesão à ferramenta, (ii) qualidade das instruções fornecidas e (iii) sucesso na resolução de exercícios.

Os resultados indicam que, dos três participantes, dois (66,7%) não seguiram completamente as instruções fornecidas, o que sugere que as orientações iniciais poderiam ser mais claras. Apenas um participante (33,3%) conseguiu seguir integralmente às instruções. Isso mostra que embora a ferramenta tenha resultados positivos, há espaço para aprimorar seu suporte a usuários com dificuldades. Em relação às dicas oferecidas pelo especialista, dois dos participantes consideraram-nas úteis, enquanto o terceiro a julgou parcialmente úteis, apontando para a necessidade de melhorias na clareza ou adaptabilidade das orientações.

Os resultados mostram boa aceitação e os relatos qualitativos reforçam que o modelo atende ao propósito inicial, mas evidenciam a necessidade de melhorias, sugerem ajustes, como incluir orientações sobre boas práticas de programação para aumentar a eficiência e o desempenho dos scripts gerados.

5. Discussão

Por não oferecer soluções prontas, o especialista incentiva o usuário a analisar o problema, identificar possíveis causas e pensar em soluções. O modelo realiza isso oferecendo orientações conceituais, ajudando no desenvolvimento da habilidade de compreender algoritmos e no incentivo ao usuário de refletir sobre suas respostas e os potenciais erros que ele pode resolver. O foco está no entendimento do problema e na construção do

algoritmo, e não na simples execução de um código copiado. Devido ao fato do modelo não oferecer soluções prontas, o mesmo coloca o usuário no centro do processo de aprendizado, transformando-o em protagonista, cumprindo seu papel como um mediador que orienta e desafia o estudante, e não simplesmente resolve o problema.

A criação do especialista foi baseado no desenvolvimento de ferramentas como o *Iris* [Bassner et al. 2024] e o *CodeHelp* [Liffiton et al. 2023] que foram elaborados para estimular um aprendizado mais ativo e reflexivo, ajudando os alunos a desenvolverem habilidades essenciais para a programação sem recorrer a respostas automáticas ou prontas. Entretanto, o modelo apresenta limitações que podem comprometer a experiência do estudante ao se comunicar com o a ferramenta; ademais, após a análise das interações, esse fator impactou somente na experiência dos usuários mais experientes, já que para os iniciantes, as explicações elucidaram dúvidas simples relacionadas à criação da resposta do problema.

Outro fator importante, é o fato do especialista ter sido criado e existir dentro da ferramenta ChatGPT 4, pois isso permite que o estudante acesse o chatbot convêncional do ChatGPT, ao inves de utilizar o chatbot personalizado (especialista). Para isso, é necessário qie os professores criem estratégias pedagógias, principalmente nos casos do especialista ser utilizado como um monitor de turma, o professor responsável precisará ressaltar que o ChatGPT 4 cria algoritmos de forma prática e eficiente, mas oferece menos oportunidades de aprendizado, enquanto o especialista foca na educação do usuário, explicando os conceitos e promovendo a autonomia. O principal diferencial do uso no modelo em sala de aula é esse suporte pedagógico contextualizado, que alinha o aprendizado aos objetivos específicos, incentivando a construção ativa do conhecimento.

6. Considerações Finais

O estudo demostra ser capaz de estimular a autonomia e promover a aprendizagem ativa, porém é fundamental equilibrar a abordagem reflexiva com soluções práticas. Isso permite atender melhor às expectativas dos usuários, independentemente do nível de experiência ou do tempo disponível para resolver o problema. Apesar de ter apresentado bons resultados iniciais, é preciso revisar periodicamente os conteúdos e estratégias e fazer atualizações que permitam corrigir problemas identificados nas interações.

Todo estudo experimental apresenta ameaças à validade, como o uso de apenas três participantes, o que limita a generalização dos resultados, e a variação nos níveis de conhecimento em programação, que pode ter influenciado as interações. Além disso, as aplicações ocorreram em um ambiente controlado, distante das condições reais de uso. O desempenho do especialista depende da qualidade do prompt, e falhas ocasionais em fornecer explicações claras ainda foram observadas. Apesar do estudo apresentar indícios que a ferramenta surtiu efeito no desenvolvimento da habilidade de compreensão de algoritmos, devido as limitações é necessario de ampliar a amostra, além disso, diversificar os contextos de aplicação e refinar o modelo para atender diferentes perfis de usuários e fortalecer a validade dos resultados. Apenar das limitações, o especialista ainda conseguiu cumprir seu papel, atuando como um mediador que auxilia o usuário na construção de sua própria resposta a problemas envolvendo programação.

Referências

[Alves et al. 2024] Alves, C., Pires, F., Melo, R., and Pessoa, M. (2024). Desenvolvimento de especialista em gamificação no chatgpt 4.0: Análise de frameworks de gamificação. In *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 1003–1014, Porto Alegre, RS, Brasil. SBC.

[Azaria 2022] Azaria, A. (2022). ChatGPT Usage and Limitations. working paper or preprint.

[Bassner et al. 2024] Bassner, P., Frankford, E., and Krusche, S. (2024). Iris: An ai-driven virtual tutor for computer science education. pages 394–400.

[da Cruz et al. 2023] da Cruz, K. R., da Silva Toledo, R., de Oliveira, A. S., Moreira, A. M., Gandin, L. R. A., et al. (2023). Ia na sala de aula: como a inteligência artificial está redefinindo os métodos de ensino. *Rebena-Revista Brasileira de Ensino e Aprendizagem*, 7:19–25.

[de Araujo 2024] de Araujo, C. A. (2024). A inteligência artificial e o desenvolvimento neuropsicológico de crianças e adolescentes. *Self-Revista do Instituto Junguiano de São Paulo*, 9:e001–e001.

[Delbone et al. 2024] Delbone, F., Wiese, I., and Silva, M. G. (2024). Análise dos efeitos do idioma na geração automática de respostas por aplicações de llm. In *Anais Estendidos do IV Simpósio Brasileiro de Educação em Computação*, pages 23–24, Porto Alegre, RS, Brasil. SBC.

[Liang et al. 2023] Liang, Y., Wang, J., Zhu, H., Wang, L., Qian, W., and Lan, Y. (2023). Prompting large language models with chain-of-thought for few-shot knowledge base question generation. *arXiv preprint arXiv:2310.08395*.

[Liffiton et al. 2023] Liffiton, M., Sheese, B. E., Savelka, J., and Denny, P. (2023). Codehelp: Using large language models with guardrails for scalable support in programming classes. pages 1–11.

[Lima 2023] Lima, J. (2023). Como o chatgpt afeta a educação e o desenvolvimento universitário. *The Trends Hub*, (3).

[Lira et al. 2024] Lira, W. A. L., dos Santos Neto, P. d. A., and Osorio, L. F. M. (2024). Uma análise do uso de ferramentas de geração de código por alunos de computação. In *Anais do IV Simpósio Brasileiro de Educação em Computação*, pages 63–71. SBC.

[Picão et al. 2023] Picão, F. F., Gomes, L. F., Alves, L., Barpi, O., and Luccheti, T. A. (2023). Inteligência artificial e educação: como a ia está mudando a maneira como aprendemos e ensinamos. *Revista Amor Mundi*, 4(5):197–201.

[Ribeiro 2023] Ribeiro, E. (2023). Do básico ao complexo: aprendendo a programar em python com o chatgpt.

[Silveira et al. 2019] Silveira, C. d., Silva, A. R. d., Herpich, F., and Tarouco, L. M. R. (2019). Uso de agente conversacional como recurso de aprendizagem sócio-educacional. *RENOTE: revista novas tecnologias na educação. Vol. 17, n. 3 (2019), 668-678.*

[Viecheneski and Carletto 2013] Viecheneski, J. P. and Carletto, M. R. (2013). Iniciação à alfabetização científica nos anos iniciais: contribuições de uma sequência didática. *Investigações em Ensino de Ciências*, 18(3):525–543.

[Vigotsky et al. 1987] Vigotsky, L. S. et al. (1987). Pensamento e linguagem.