

1ª edição

Aprendendo a Programar com a Ajuda de IAs: Estratégias para Turbinar Seus Estudos



Tiago Roberto Kautzmann
Fabrício Malta de Oliveira

Este livro também está disponível para leitura no Kindle.
Encontre-o na Amazon ([amazon.com.br](https://www.amazon.com.br)) pesquisando pelo título.

**Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)**

Kautzmann, Tiago Roberto

Aprendendo a programar com a ajuda de IAs [livro eletrônico] : estratégias para turbinar seus estudos / Tiago Roberto Kautzmann, Fabrício Malta de Oliveira. -- 1. ed. -- União da Vitória, PR : Ed. dos Autores, 2025. -- (Série aprendizagem de programação)

PDF

Bibliografia.

ISBN 978-65-01-36535-0

1. Ciência da Computação 2. Inteligência artificial 3. Linguagem de programação para computadores - Estudo e ensino I. Oliveira, Fabrício Malta de. II. Título. III. Série.

25-257474

CDD-005.133

Índices para catálogo sistemático:

1. Linguagem de programação : Computadores : Processamento de dados 005.133

Eliane de Freitas Leite - Bibliotecária - CRB 8/8415

Com o objetivo de aprimorar a clareza e a qualidade dos textos apresentados neste livro, utilizamos o ChatGPT como ferramenta auxiliar na revisão e no refinamento dos textos.

Sobre os autores

Tiago Roberto Kautzmann



Atualmente atua como Professor do Instituto Federal do Paraná (IFPR). Doutor em Computação Aplicada pela Unisinos - Universidade do Vale do Rio dos Sinos (2018 - 2022), com período sanduíche realizado em Marselha (França), junto à Aix-Marseille Université (2019). Mestre em Computação Aplicada pela Unisinos (2013 - 2015). Tecnólogo em Sistemas para Internet pela Universidade Feevale (2009 - 2011). Formação pedagógica pelo Programa Especial de Formação Pedagógica da Universidade Feevale (2012 - 2013). Realiza pesquisa científica em Inteligência Artificial aplicada à Educação. Professor há mais de 20 anos, ministra disciplinas sobre algoritmos, ciência de dados, inteligência artificial, estruturas de dados, programação orientada a objetos, desenvolvimento de software, banco de dados e lógica.

Contato: tkautzmann@gmail.com

Fabrício Malta de Oliveira



Atualmente atua como Professor do Instituto Federal do Paraná (IFPR). Mestrando em Ciência da Computação pela UFSCar - Universidade Federal de São Carlos. Possui graduação em Engenharia da Computação pelo Instituto Federal de São Paulo (IFSP), Campus Birigui, com pós-graduação em Análise de Dados. Possui experiência profissional em desenvolvimento web semântico e responsivo. Estuda atualmente sobre metaverso, realidade virtual e realidade estendida voltada ao uso de times de software.

Contato: fabricio.malta@outlook.com

Prefácio

Nos últimos anos, a Inteligência Artificial tem transformado diversas áreas do conhecimento, e o ensino de programação não é exceção. Ferramentas de IA generativa, como o ChatGPT passaram a auxiliar programadores iniciantes e experientes, fornecendo sugestões de código, explicando conceitos e agilizando o processo de desenvolvimento de software. No entanto, o impacto dessas tecnologias no aprendizado de alunos ainda está em discussão, pois, se utilizadas sem critério, podem levar à dependência excessiva e à superficialidade no entendimento dos conceitos fundamentais de programação.

Neste livro, buscamos explorar estratégias para que os alunos se beneficiem das IAs generativas na sua aprendizagem de programação. Aprender a programar é uma tarefa desafiadora, que exige compreensão de lógica, sintaxe, estrutura de dados e resolução de problemas. Dessa forma, o uso consciente das IAs deve ser pensado como um suporte ao aprendizado ativo, não como um substituto ao raciocínio e à prática.

Sabemos que essas ferramentas continuarão a evoluir e que, no momento em que você, leitor, estiver lendo este livro, novas tecnologias podem surgir, enquanto outras podem ter sido descontinuadas ou aprimoradas. Nossa objetivo é apresentar estratégias que você possa

utilizar de maneira crítica e produtiva, independentemente das inovações tecnológicas que venham a surgir.

Acreditamos que a chave para uma aprendizagem eficiente com o uso de IAs generativas está na forma como o estudante interage com essas ferramentas. Não se trata de pedir respostas prontas a essas IAs, mas sim de fazer perguntas inteligentes, explorar soluções alternativas, entender erros e aprimorar o próprio processo de pensamento computacional. Ao longo deste livro, apresentamos diversas estratégias para que os alunos possam potencializar seus estudos com IAs generativas, garantindo que o aprendizado de programação aconteça de maneira significativa e duradoura.

Por fim, convidamos você, leitor, a refletir e experimentar as estratégias propostas. Queremos que este livro seja mais do que um conjunto de informações: que ele sirva como um guia prático e dinâmico para o seu aprendizado de programação com o auxílio das IAs generativas. Ficaremos felizes em receber seus feedbacks sobre como essas abordagens impactaram sua experiência nos estudos de programação e como a IA pode continuar sendo uma aliada poderosa nesse processo.

Os Autores.

Conteúdo

1 Introdução.....	7
2 Aprender a programar é desafiador, e está tudo bem!.....	15
3 IAs generativas.....	19
4 A chave do sucesso: escrever bons prompts.....	22
5 As IAs generativas podem alucinar!.....	25
6 Ferramentas de IA generativa.....	30
7 O uso de IAs generativas em empresas de desenvolvimento de software.....	37
8 O uso de IAs generativas na aprendizagem.....	42
9 Riscos de usar IAs generativas na aprendizagem de programação.....	46
10 Estratégias para o aluno se beneficiar das IAs generativas na aprendizagem de programação.....	49
Estratégia 1.....	52
Estratégia 2.....	54
Estratégia 3.....	56
Estratégia 4.....	58
Estratégia 5.....	60
Estratégia 6.....	62
Estratégia 7.....	64
Estratégia 8.....	66
Estratégia 9.....	68
Estratégia 10.....	70
11 Considerações sobre as estratégias.....	72
12 O futuro do uso das IAs generativas na aprendizagem de programação.....	73
13 Mas e agora? As IAs substituirão os programadores? Devo continuar estudando programação?.....	75
Referências.....	77

1 Introdução

A aquisição de habilidades de Pensamento Computacional (PC) tem sido considerada tão importante quanto adquirir habilidades de leitura, escrita e aritmética. Habilidades de PC envolvem processos cognitivos como pensamento lógico e sistemático e seleção de algoritmos para resolução de problemas em diversos contextos, não apenas no campo profissional da computação (DENNY et al., 2024b; TSAI; WANG; HSU, 2019). Estas habilidades têm sido incluídas nas bases curriculares de alguns países, como Inglaterra (DEPARTMENT FOR EDUCATION, 2014), Estados Unidos (NATIONAL SCIENCE FOUNDATION, [s.d.]) e o Brasil (MINISTRO DA EDUCAÇÃO, 2022).

Aprender a programar é uma forma eficiente de adquirir habilidades de Pensamento Computacional (ATMATZIDOU; DEMETRIADIS, 2016; KAZIMOGLU et al., 2012), além de ser fundamental para atuar em diferentes campos de trabalho na área de Tecnologia da Informação (TI), como em desenvolvimento de software. Estas habilidades também têm se mostrado cada vez mais requeridas em diversas outras áreas de atuação, como nas engenharias, Matemática e Física. No entanto, estudos mostram que o ensino de programação é um desafio para os currículos escolares. Com frequência, estudantes de

disciplinas introdutórias de programação se deparam com dificuldades na compreensão de conceitos importantes (LOKSA et al., 2016). Estudos já reportaram uma taxa média de evasão e insucesso em cursos introdutórios de programação de cerca de 50% (BENNEDSEN; CASPERSEN, 2007, 2019).

Os desafios inerentes às atividades de programação de computadores não são percebidos apenas no ambiente escolar, mas também nas empresas de desenvolvimento de software. Com o objetivo de atenuar problemas decorrentes da complexidade de desenvolver software, empresas têm adotado o método de programação em pares (*pair programming*), na qual dois programadores trabalham juntos em uma mesma estação de trabalho. Um dos programadores, o "piloto", escreve o código, enquanto o outro programador, o "navegador", revisa e sugere melhorias (BIRD et al., 2023; WILLIAMS; KESSLER, 2002). Esta prática melhora a qualidade do código, reduz erros, e possibilita o suporte imediato entre os pares (COCKBURN; WILLIAMS, 2000). A programação em pares também tem sido utilizada em atividades de ensino de programação, e tem apresentado resultados promissores ao proporcionar uma experiência colaborativa que simula o ambiente de trabalho. A interação entre os estudantes facilita a compreensão de conceitos, a resolução de problemas e a detecção de erros. Além de melhorar o aprendizado, essa abordagem também aumenta a motivação dos alunos, aliviando a sensação de isolamento que ocorre na

aprendizagem individual (SALLEH; MENDES; GRUNDY, 2010; UMAPATHY; RITZHAUPT, 2017). Apesar dos seus benefícios, a programação em pares também enfrenta alguns desafios, como uma eventual incompatibilidade entre os parceiros em termos de conhecimento e personalidade (SALLEH; MENDES; GRUNDY, 2010). Diferenças significativas entre os parceiros podem comprometer a colaboração e o aprendizado. Há o risco de um parceiro dominar o processo e limitar a participação do outro. Também pode surgir problemas como discriminação, preocupação com o desempenho individual e a divisão desigual do trabalho. A implementação desta organização de trabalho exige atenção a esses fatores.

A partir de 2022 observamos o advento dos grandes modelos de linguagem (*Large Language Models*, ou LLMs) em ferramentas como o ChatGPT, o Claude e o Gemini. Um LLM é uma inteligência artificial gerativa, ou seja, um tipo de IA que é treinada com grandes volumes de dados e que é capaz de gerar textos, imagens, músicas e outros conteúdos coerentes ao contexto. A geração desses conteúdos ocorre a partir de uma entrada, o *prompt*, que serve como instrução ou estímulo para que a IA gere o conteúdo. Um exemplo de prompt é o seguinte: "Explique os principais fatores que levaram à Proclamação da República no Brasil em 1889, destacando o papel dos militares, a influência das ideias republicanas, e as crises econômicas e sociais que marcaram o final do Império.". Os *prompts* orientam a direção e o contexto das

respostas das IAs generativas para que atendam às expectativas do usuário (WHITE et al., 2023).

Estas IAs também têm sido amplamente usadas em atividades de programação de computadores. Elas são capazes de gerar códigos relevantes ao contexto do prompt, tornando-se uma ferramenta valiosa nas atividades de desenvolvimento de software (PORTER; ZINGARO, 2024). Estes modelos LLMs têm sido usados em ferramentas de apoio ao desenvolvimento de software como o GitHub Copilot e o Amazon Q. Elas auxiliam o programador nas suas tarefas, como gerar soluções de código, identificar erros e sugerir melhorias nos códigos, e estão impactando o mercado de desenvolvimento de software. Uma pesquisa com desenvolvedores mostrou que o uso do GitHub Copilot está gerando até 46% dos códigos e ajudando os desenvolvedores a codificar até 55% mais rápido (DOHMKE, 2024). Além disso, uma pesquisa com mais de 15 mil programadores brasileiros (CÓDIGO FONTE TV, 2024) mostrou que 83,61% dos programadores participantes da pesquisa utilizam IA no trabalho. Em torno de 88% dos desenvolvedores sentem-se mais produtivos usando IAs, totalmente (62,69%) ou parcialmente (25,65%). A pesquisa também encontrou que em torno de 80% dos programadores utilizam o ChatGPT (66,65%) e o GitHub Copilot (13,51%).

O uso das IAs generativas nas empresas de software e no ensino utilizam uma abordagem similar ao de programação em pares humanos,

que é a chamada programação em pares IA-humano (*AI pair programming*). A IA atua como o parceiro "navegador" do programador humano (o "piloto"), oferecendo assistência e sugestões. Esta abordagem preserva os benefícios da colaboração e revisão do código e evita conflitos envolvendo discrepâncias de conhecimento e conflitos de personalidade entre humanos (MA; WU; KOEDINGER, 2023a). O ChatGPT e o GitHub Copilot têm sido usados para a programação em pares IA-humano, inclusive no contexto de aprendizagem de programação de computadores.

Apesar do recente advento das IAs generativas que utilizam modelos LLM, já foram encontradas evidências que o uso dessas ferramentas no ensino pode beneficiar a aprendizagem de programação mas também pode prejudicar quando mal utilizadas (PRATHER et al., 2023). O GitHub Copilot é uma ferramenta que fornece exemplos de códigos e sugestões em tempo real (BIRD et al., 2023; MA; WU; KOEDINGER, 2023a). Já foram encontradas evidências que estudantes iniciantes de disciplinas introdutórias de programação podem aceitar sugestões da ferramenta sem compreender a lógica subjacente, resultando em aprendizado superficial, dependência excessiva da ferramenta (BECKER et al., 2023; DENNY et al., 2024b; PRATHER et al., 2023), e uma degradação de habilidades de codificação (DENNY et al., 2024a). É importante também destacar que o GitHub Copilot não é uma ferramenta desenvolvida especificamente para o contexto de

aprendizagem, de forma que seu comportamento não é adaptado às necessidades de um estudante.

Considerando a forte presença dessas ferramentas de IA no mercado de trabalho de desenvolvimento de software, não é possível pensar em uma formação de aluno nesta área sem considerar a aquisição de habilidades relacionadas ao uso dessas IAs. Novas abordagens pedagógicas são necessárias para desenvolver estas novas habilidades da era da IA generativa (DENNY et al., 2023). Além de ser cada vez mais importante ensinar os alunos a ler e a entender códigos gerados pelas IAs, uma habilidade igualmente importante é a de formular *prompts* efetivos para gerarem uma resposta satisfatória (DENNY et al., 2024b). Além disso, aprender a se comunicar melhor com essas IAs pode também ajudar o aluno no desenvolvimento e fortalecimento de suas habilidades de programação (GHIMIRE; EDWARDS, 2024).

O que vou aprender neste livro?

Neste livro, os autores apresentam 10 estratégias de utilização das IAs generativas que podem ser adotadas durante o processo de aprendizagem em programação. As estratégias foram elaboradas a partir da experiência prática dos autores, para serem adotadas por alunos no aprendizado de programação, tanto em aulas presenciais quanto em cursos à distância. Embora ainda não haja comprovação quantitativa ou

qualitativa da eficácia e da força dessas estratégias para a aprendizagem dos alunos, os autores acreditam que essas estratégias podem contribuir significativamente para o ensino, enquanto aguardam os resultados de pesquisas em andamento na área de Informática aplicada à Educação. Os autores consideram importante o advento de pesquisas que tenham como objetivo desenvolver abordagens pedagógicas que auxiliem os alunos a se comunicarem melhor com essas IAs e tirarem proveito delas para melhorar sua aprendizagem e o processo de desenvolvimento de software.

Além de apresentar estas estratégias, este livro traz diversos apontamentos sobre as IAs generativas, como as práticas de comunicação com elas através de prompts, as ferramentas atualmente mais relevantes e os impactos delas no mercado de trabalho e no processo de aprendizagem em programação. O texto também discute os riscos associados ao uso dessas tecnologias na educação e oferece uma visão prospectiva sobre seu futuro. Por fim, o livro busca responder a uma das dúvidas mais recorrentes entre os alunos: as IAs substituirão os programadores? Os alunos devem continuar aprendendo programação?

A quem se destina este livro?

Este livro destina-se a estudantes e educadores que buscam melhorar a aprendizagem e o ensino da programação, seja em ambientes

presenciais ou em cursos a distância. Ao reunir estratégias desenvolvidas com base em experiências práticas dos autores, o livro oferece insights para quem deseja superar os desafios tradicionais do ensino da programação. Assim, ele se propõe a ser uma ferramenta útil tanto para profissionais da educação que procuram diversificar suas metodologias quanto para alunos que desejam aprofundar sua compreensão e desenvolver habilidades essenciais de codificação. Profissionais da área de desenvolvimento de software também poderão encontrar discussões interessantes para suas práticas.

Como os estudantes devem estudar com este livro?

Para estudar com este livro, os autores recomendam que o estudante adote uma abordagem ativa e reflexiva. Que ele explore cada uma das 10 estratégias apresentadas, considerando como elas podem ser aplicadas ao seu contexto de aprendizado, seja em sala de aula ou em cursos a distância. Ao longo da leitura, o aluno deve refletir sobre as experiências práticas dos autores e como essas podem ser adaptadas às suas próprias necessidades e desafios. É importante também, ao aplicar as estratégias, praticar regularmente o que foi aprendido, ajustando os métodos conforme necessário para otimizar sua aprendizagem.

2 Aprender a programar é desafiador, e está tudo bem!

Aprender a programar é uma tarefa desafiadora! E sim, querido aluno, se você está se sentindo confuso em algum nível, saiba que isso é completamente normal! Nós, autores, já “quebramos muito a cabeça”, já ficamos muito confusos e muito irritados quando passávamos 2 horas tentando descobrir um erro no código até perceber que faltava um simples ponto-e-vírgula. Quem nunca, não é mesmo? Se você está encontrando obstáculos pelo caminho, se fica confuso ao programar, saiba que você não está sozinho - pelo contrário, isso faz parte do processo de aprendizado. Cada erro, cada dúvida e cada problema resolvido são passos essenciais para o seu crescimento como programador. Então, siga em frente, pois você está no caminho certo! E nós, programadores mais experientes, também nos deparamos com muita confusão e ficamos trancados muitas vezes. Principalmente quando precisamos resolver tipos de problemas que nós nunca havíamos resolvido algo similar antes. Faz parte do game, e está tudo bem! Que bom que nas empresas temos colegas e líderes técnicos para nos ajudar. Temos também o Google, o Stack Overflow e, nos dias atuais,

ferramentas de IA generativa como o ChatGPT. Nós, definitivamente, não estamos sozinhos nesta jornada de programação.

Aprender a programar envolve complexidades! Além de precisar entender o problema subjacente da tarefa de programação que é atribuída pelo professor, o aluno também precisa mobilizar esforços cognitivos para identificar e relacionar conhecimentos e estratégias de resolução de problemas e tomar decisões lógicas (CHI; OHLSSON, 2005; LEHMAN et al., 2013).

Aprender a programar exige do aluno uma combinação de habilidades cognitivas, lógicas e analíticas. Primeiramente, compreender o problema proposto é uma etapa fundamental, pois envolve a interpretação correta dos requisitos e das restrições da tarefa de programação. Esse entendimento inicial é essencial para que o estudante possa definir uma abordagem eficaz para a solução. Além disso, a programação exige que o aluno mobilize diversos conhecimentos adquiridos ao longo de seu aprendizado, como conceitos de lógica, estruturas de dados, algoritmos e sintaxe da linguagem de programação utilizada.

Outro fator crítico nesse processo é a necessidade de desenvolver estratégias para a resolução de problemas. Isso envolve a capacidade de decompor problemas complexos em partes menores e mais gerenciáveis, escolher as melhores abordagens para cada situação e implementar soluções eficientes. Além disso, a programação exige tomadas de

decisão constantes, onde o aluno precisa avaliar diferentes caminhos possíveis, antecipar erros e otimizar seu código.

A complexidade do aprendizado torna o ensino de programação um grande desafio. Neste contexto, torna-se importante a adoção pelos professores de abordagens pedagógicas que facilitem a assimilação dos conceitos e incentivem a prática ativa dos estudantes, como o aprendizado baseado em projetos, a programação em pares e, nos dias atuais, o uso de ferramentas de inteligência artificial como apoio no desenvolvimento das habilidades de codificação.

Um exemplo prático de tarefa que reflete a complexidade do processo de aprendizado de programação poderia ser o desenvolvimento de um algoritmo para ordenar de forma eficiente uma lista de números. Essa tarefa exige que o aluno, primeiramente, entenda o problema (ou seja, que ele compreenda que precisa ordenar uma coleção de números de maneira crescente ou decrescente). Em seguida, o aluno deve mobilizar seus conhecimentos sobre estruturas de dados, como listas ou arrays, e algoritmos de ordenação, como o Bubble Sort, Quick Sort e Merge Sort. Depois deve escolher a abordagem mais adequada com base nas características do problema, como o tamanho da lista e os requisitos de eficiência. Não é uma tarefa tão simples.

Além de entender o problema e modelar uma solução, o aluno também precisa escrever o código em uma linguagem de programação (como Python ou Java), implementando a lógica de ordenação e lidando

com erros que podem surgir durante a codificação e a execução, como listas vazias ou entradas não numéricas do teclado do usuário. Durante o processo, ele também pode precisar debugar o código, testar diferentes entradas e verificar se o algoritmo funciona corretamente. Cada uma dessas etapas exige esforço cognitivo para identificar soluções, aplicar estratégias e avaliar alternativas de forma lógica e eficiente.

Esse exemplo destaca como a programação vai além da simples escrita de código, envolvendo habilidades de resolução de problemas, tomada de decisões e aplicação de conceitos teóricos e práticos.

Nos próximos capítulos vamos abordar como as IAs generativas funcionam, como elas são utilizadas no mercado de desenvolvimento de software e como elas podem ajudar os alunos nesta complexa tarefa que é aprender a programar.

3 IAs generativas

Nos últimos anos, as IAs generativas se tornaram um tema amplamente discutido, não apenas entre especialistas em Ciência da Computação, mas também por pessoas de diversas áreas e pelo público em geral. Esse aumento no interesse se deve, principalmente, ao avanço e popularização de ferramentas como ChatGPT, DeepSeek, Sora, Midjourney, Dall E, Grok e GitHub Copilot, que demonstraram a capacidade da IA de gerar textos, imagens, códigos e até mesmo simular interações humanas de forma convincente. O impacto dessas tecnologias no mercado de trabalho, na educação e na criatividade despertou debates sobre ética, automação, produtividade e o futuro das profissões. A acessibilidade dessas ferramentas, que antes eram restritas a pesquisadores e desenvolvedores, também contribuiu para essa ampla disseminação do tema, tornando a IA generativa um fenômeno global que transcende os limites da computação.

Estas ferramentas de IAs generativas como o ChatGPT funcionam utilizando grandes modelos de linguagem (LLMs) treinados com vastas quantidades de texto e imagens para gerar respostas coerentes e apropriadas ao contexto. Esses modelos são baseados em redes neurais profundas, especialmente a arquitetura *Transformer*, que aprende

padrões, relações semânticas e estruturas linguísticas a partir dos dados de treinamento. Quando recebem um *prompt* de entrada, essas IAs geram texto prevendo a sequência de palavras mais provável, levando em conta o contexto explícito e implícito contido no *prompt* e as informações previamente aprendidas. Esse processo permite que elas gerem respostas, histórias, códigos e imagens de maneira fluida, relevante e consistente (LOUKIDES, 2023).

O conceito de *prompt* é fundamental para o entendimento dessas IAs. O *prompt* é o comando ou entrada fornecida a um sistema de IA para iniciar uma resposta. Em termos simples, é a pergunta, instrução ou frase que você digita para a IA gerar uma resposta. O *prompt* orienta o modelo de IA sobre o tipo de conteúdo ou a tarefa esperada, influenciando diretamente o tipo de resposta que será gerada. Por exemplo, ao pedir "Explique o conceito de estrutura de seleção em programação de computadores.", essa frase serve como *prompt* para a IA gerar uma explicação relevante (LOUKIDES, 2023).

Um estudo (GHIMIRE; EDWARDS, 2024) encontrou evidências sobre os tipos de *prompts* que os alunos mais geram como entrada para a IA enquanto realizam tarefas de programação, a saber: a) ajuda na depuração de código: os *prompts* dos alunos pedem ajuda para a IA identificar ou corrigir erros em um trecho de código; b) geração de trecho de código: os *prompts* pedem para a IA gerar uma parte específica do código, como uma função no JavaScript; c) perguntas conceituais: os

prompts dos alunos pedem para a IA alguma explicação sobre conceitos ou sobre um trecho do algoritmo, ao invés de pedir para a IA gerar o código; d) solução completa: *prompts* que pedem à IA uma solução completa para um exercício de programação.

Quando os alunos pedem à IA por uma solução completa para um exercício atribuído pelo professor, é comum eles utilizarem como *prompt* os próprios textos dos enunciados de exercícios disponibilizados pelo professor. Estudos recentes mostraram que apenas usar esses enunciados nem sempre é suficiente para uma LLM gerar um código correto (FINNIE-ANSLEY et al., 2022, 2023). Geralmente é necessária uma modificação manual dos *prompts* de forma a incluir informações algorítmicas explícitas (TANG et al., 2022). Mais adiante, neste livro, discutiremos os riscos de aprendizagem para os alunos quando eles pedem soluções completas para as tarefas.

Escrever prompts exige algum conhecimento do usuário no que chamamos de engenharia de prompts, ou seja, um conjunto de habilidades para uma comunicação efetiva do usuário para com a IA generativa (WHITE et al., 2023). No próximo capítulo, nós apresentamos sugestões sobre como escrever bons prompts.

4 A chave do sucesso: escrever bons prompts

Os prompts são as instruções ou comandos fornecidos pelos usuários para interagir com as IAs gerativas, orientando-as na geração de respostas relevantes e contextualizadas. Eles podem variar de simples perguntas a descrições detalhadas de tarefas, influenciando diretamente a qualidade e a precisão do conteúdo gerado. Em essência, o prompt funciona como a entrada para o modelo de IA, que, com base nele, processa padrões e estatísticas extraídos dos dados de treinamento para produzir uma resposta. Quanto mais claro e específico for o prompt, maior a probabilidade de obter uma resposta útil e alinhada às expectativas do usuário.

Exemplo de um prompt:

"Explique o conceito de programação orientada a objetos de forma clara e objetiva, usando exemplos práticos em Python. Inclua uma explicação sobre classes, objetos, herança e polimorfismo."

A chave do sucesso ao usar IAs gerativas está na arte de escrever bons prompts (DENNY et al., 2024a). Prompts bem estruturados

permitem obter respostas mais precisas, criativas e alinhadas ao objetivo desejado. Para isso, é essencial utilizar descrições detalhadas, fornecer contexto suficiente e, quando necessário, especificar o formato esperado da resposta. Além disso, a experimentação e o refinamento contínuo dos prompts ajudam a melhorar a interação com a IA, garantindo que ela gere conteúdos mais adequados e eficazes para cada situação. Dessa forma, dominar a escrita de prompts é uma habilidade fundamental para maximizar os benefícios das IAs geratativas.

A seguir são apresentadas sugestões para escrever bons prompts. As sugestões apresentadas aqui foram sugeridas pelo próprio ChatGPT. Quem melhor que o próprio ChatGPT para explicar como melhor se comunicar com ele, não é mesmo?

Seja claro e específico

Quanto mais claro e detalhado for o prompt, melhor será a resposta. Evite ambiguidades e descreva exatamente o que você quer, incluindo contexto, resultados esperados ou perguntas diretas. Por exemplo, em vez de "Explique programação", prefira "Explique o conceito de loops em Python para iniciantes". O prompt delimita claramente o assunto a ser abordado, indica que o público-alvo são "iniciantes", e também especifica o que se espera — uma explicação do conceito de loops em Python.

Defina o formato da resposta

Se quiser um tipo específico de resposta, como uma lista, exemplo prático ou passo a passo, mencione isso no prompt. Exemplo: "Liste cinco melhores práticas de segurança em desenvolvimento web".

Use perguntas abertas para obter mais detalhes: Use perguntas abertas para obter mais detalhes: Evite perguntas que resultem em respostas de 'sim' ou 'não'. Perguntas abertas, como "Quais são os desafios da implementação de metodologias ágeis?", incentivam respostas mais detalhadas.

Itere e refine: Se a resposta não for o que você espera, refine o prompt, melhore ele com mais detalhes. O ChatGPT pode melhorar a precisão com pequenos ajustes no seu prompt.

No próximo capítulo apresentamos uma discussão sobre um ponto de atenção ao usar essas ferramentas: **elas podem alucinar!** Mesmo sendo tecnologias avançadas e aparentemente confiáveis, as IAs gerativas podem gerar informações incorretas ou apenas parcialmente incorretas. Entender esse fenômeno é fundamental para usá-las com consciência e melhor aproveitar suas potencialidades.

5 As IAs generativas podem alucinar!

Mas o que é uma alucinação no contexto de um software de IA? Até hoje você achava que apenas um humano poderia alucinar, certo? Uma alucinação ocorre quando a IA gera uma resposta aparentemente realista, mas que, na verdade, não é uma informação factual, é sem sentido ou inconsistente com o prompt fornecido (WALDO; BOUSSARD, 2025). Vamos a um exemplo de uma IA generativa alucinando:

Prompt do usuário:

Quem ganhou o Oscar de Melhor Atriz em 2023?

Resposta alucinada da IA:

A atriz Suzana Vieira ganhou o Oscar de Melhor Atriz em 2023 pelo filme Paixão Infinita.

Mas por que esta resposta é uma alucinação?

- A atriz Suzana Vieira não ganhou o Oscar em 2023.
- A atriz Suzana Vieira não estrelou nenhum filme com este título.

- A IA gerou uma resposta que parece realista, principalmente para pessoas que não acompanham notícias de cinema, mas que, na verdade, é FALSA.

Alucinações podem ocorrer para assuntos de diferentes áreas, como a IA inventar referências científicas, códigos errados ou até respostas imprecisas sobre fatos históricos. É por isso que a validação humana e a verificação das informações são essenciais ao usar IAs generativas.

Os modelos de IA generativa como o ChatGPT são desenvolvidos por meio de técnicas computacionais de aprendizado de máquina. Estas técnicas são aplicadas a grandes quantidades de dados, oriundos principalmente da internet. Durante o treinamento, essas IAs processam enormes volumes de informação e ajustam um conjunto massivo de parâmetros - que podem chegar a trilhões - para **capturar as probabilidades de quais palavras deveriam aparecer em uma sequência de palavras** (WALDO; BOUSSARD, 2025). Ou seja, essas IAs são especialmente eficazes em concatenar palavras de maneira coerente com o contexto fornecido. A previsão da próxima palavra numa sequência de palavras baseia-se nas probabilidades de ocorrência observadas nos dados de treinamento, ou seja, baseados nos textos que a IA já leu. Quando um desses modelos de IA afirma que "a grama é verde", ele não faz isso porque entende esse fato sobre o mundo real,

mas porque "a grama é" é frequentemente seguido por "verde" nos dados em que foi treinado - nos textos que ela leu anteriormente. Assim, a geração de texto não é baseada em compreensão genuína, mas sim em padrões estatísticos extraídos de textos vistos anteriormente pela IA (WALDO; BOUSSARD, 2025).

No entanto, quanto mais obscuro ou controverso for o assunto, maior a probabilidade de que a IA gere respostas incorretas. Isso ocorre porque a base de treinamento pode conter informações contraditórias, insuficientes ou tendenciosas, levando a respostas que podem parecer confiáveis, mas que não necessariamente refletem a realidade (WALDO; BOUSSARD, 2025).

A propagação de fake news é um dos riscos mais preocupantes associados às alucinações de IAs generativas. Como esses modelos não possuem um entendimento real do mundo e baseiam suas respostas em padrões estatísticos de palavras, eles podem gerar informações falsas que parecem plausíveis e bem fundamentadas. Se esses textos forem compartilhados sem verificação, podem contribuir para a disseminação de desinformação em larga escala (WALDO; BOUSSARD, 2025).

Um problema crítico é que as alucinações da IA podem surgir em tópicos sensíveis, como política, saúde e ciência. Por exemplo, se um modelo gerar um artigo falso sobre um tratamento médico não comprovado e ele for compartilhado como se fosse verdadeiro, isso pode levar pessoas a tomarem decisões prejudiciais. Da mesma forma, no

contexto político, uma IA pode fabricar declarações ou eventos que nunca aconteceram, influenciando opiniões e potencialmente impactando processos democráticos.

Como os textos gerados são frequentemente bem estruturados e escritos em um tom confiante, eles podem parecer mais confiáveis do que realmente são. Isso aumenta o risco de que leitores desatentos ou menos experientes na checagem de fontes tomem essas informações como verdadeiras. Por isso, é fundamental que usuários e plataformas adotem medidas de verificação, como a checagem cruzada com fontes confiáveis e o uso de ferramentas especializadas na detecção de desinformação.

Apesar dos avanços na inteligência artificial e no processamento de linguagem natural, muitas ferramentas desenvolvidas para detectar desinformação (fake news) ainda apresentam eficiência limitada, o que tem gerado desapontamento entre pesquisadores e usuários. Essas ferramentas enfrentam desafios complexos, como a dificuldade de interpretar o contexto, detectar ironia ou sarcasmo e lidar com informações parcialmente verdadeiras que são manipuladas para enganar. Além disso, os modelos de IA podem ser influenciados por viéses algorítmicos, resultando em falsos positivos ou falsos negativos, comprometendo sua confiabilidade. A rápida evolução das estratégias de desinformação também dificulta a atualização dessas ferramentas, tornando-as menos eficazes na identificação de conteúdos enganosos.

Esse cenário evidencia que, embora a tecnologia possa ser uma aliada na luta contra as fake news, a verificação humana e o pensamento crítico ainda são indispensáveis.

É importante reforçar que as IAs generativas não são apenas aquelas que geram texto, mas também as que geram imagens e vídeos, como o próprio ChatGPT é capaz de fazer. Essa capacidade de criar imagens e vídeos hiper-realistas coloca em xeque a credibilidade do que vemos, fazendo com que a tradicional expressão "**é ver pra quer**" adquira uma nova conotação: não basta mais apenas ver, é crucial verificar e validar a autenticidade dos conteúdos. Esse cenário demanda uma abordagem mais crítica e rigorosa, tanto por parte dos consumidores de informação quanto dos reguladores, para que se possa mitigar os riscos de desinformação e preservar a integridade da comunicação na era digital.

6 Ferramentas de IA generativa

Neste capítulo, exploraremos as principais ferramentas de inteligência artificial generativa que têm se destacado como auxiliares poderosos no campo da programação de computadores. A IA generativa, caracterizada por sua capacidade de gerar conteúdos, soluções e sugestões a partir de dados e prompts fornecidos, tem se tornado uma aliada indispensável para desenvolvedores e equipes de software. Ferramentas como GitHub Copilot, ChatGPT, e outras têm transformado a maneira como os programadores abordam tarefas do cotidiano, desde a escrita de código até a resolução de problemas complexos, passando pela documentação e otimização de processos. Descrevemos algumas dessas ferramentas, suas funcionalidades e como elas podem ser integradas ao fluxo de trabalho de programadores para aumentar a produtividade, qualidade do código e a aprendizagem de conceitos complexos.

ChatGPT (<https://chatgpt.com>): Possivelmente é a ferramenta mais conhecida de todas. Embora o ChatGPT não seja uma ferramenta projetada exclusivamente para programação, sua capacidade de gerar texto em linguagem natural pode ser extremamente útil para programadores. Programadores frequentemente utilizam o ChatGPT para discutir problemas de código, entender conceitos complexos de

programação, pedir explicações de mensagens de erro e até mesmo gerar documentação técnica. Sua flexibilidade também permite que seja usado para ajudar na geração de exemplos de código ou para fornecer sugestões sobre como resolver problemas de lógica. Além disso, o ChatGPT pode ser uma ferramenta útil para refinar a comunicação escrita, como a criação de README files, documentação de software ou até mesmo para gerar ideias sobre como melhorar a usabilidade e design de sistemas.

GitHub Copilot (<https://github.com/features/copilot>):

Desenvolvido pela GitHub em parceria com a OpenAI, é uma das ferramentas de IA generativa mais populares para auxiliar programadores. O GitHub Copilot pode ser usado em diversas IDEs, incluindo Visual Studio Code e as IDEs da JetBrains (como IntelliJ IDEA e PyCharm), permitindo que os desenvolvedores recebam sugestões de trechos de código diretamente em seus ambientes de desenvolvimento preferidos. Ele funciona como um assistente de codificação que sugere trechos de código enquanto o programador está escrevendo, com base no contexto do código existente. O Copilot é capaz de compreender e gerar código em várias linguagens de programação, ajudando os desenvolvedores a escrever código mais rápido e a superarem bloqueios criativos. Além disso, ele pode sugerir

soluções para problemas comuns, melhorar a legibilidade do código e até mesmo oferecer implementações alternativas para uma mesma tarefa.

Cursor (<https://www.cursor.com>): O Cursor é uma ferramenta de inteligência artificial voltada para auxiliar desenvolvedores na escrita de código, oferecendo funcionalidades avançadas como autocompletar inteligente, geração de código baseada em contexto e explicação de trechos de código. Integrada a editores como o VS Code, a ferramenta utiliza modelos de IA para sugerir otimizações, corrigir erros e até mesmo gerar documentação automaticamente. O Cursor também permite interação conversacional, possibilitando que programadores façam perguntas diretamente dentro do ambiente de desenvolvimento e recebam respostas contextualizadas. Seu objetivo é aumentar a produtividade dos desenvolvedores, reduzir o tempo gasto com debugging e facilitar o aprendizado de novas tecnologias e linguagens de programação.

OpenAI Codex (<https://openai.com/index/openai-codex>): O OpenAI Codex é um modelo de IA generativa especializado em programação, sendo a base do GitHub Copilot. Ele é treinado em uma vasta quantidade de código-fonte público e pode compreender e gerar código em diversas linguagens, como Python, JavaScript, Java, C++, entre outras. O Codex também auxilia programadores ao sugerir trechos

de código, completar funções automaticamente e gerar código a partir de descrições em linguagem natural. Sua capacidade de entender comandos em linguagem natural facilita a prototipagem rápida e a automação de tarefas repetitivas. Além disso, ele pode ser utilizado por meio da API da OpenAI, permitindo integração em diferentes ambientes e fluxos de trabalho. Se o objetivo for desenvolvimento diário dentro de uma IDE, o GitHub Copilot é a melhor escolha, pois é otimizado para produtividade e usabilidade imediata. Já se a necessidade for construir aplicações que utilizam IA para gerar código sob demanda ou integrar IAs em outras plataformas, o OpenAI Codex via API é mais indicado.

Amazon Q (<https://aws.amazon.com/q>): É uma ferramenta de IA generativa da AWS (Amazon Web Services) projetada para auxiliar desenvolvedores e equipes de software no processo de programação. Integrado ao ecossistema da AWS, ele pode ser utilizado para geração de código, otimização de soluções, depuração e explicação de trechos de código em diversas linguagens de programação. Além disso, o Amazon Q é capaz de responder perguntas técnicas, fornecer exemplos práticos e sugerir boas práticas de desenvolvimento com base nos serviços da AWS. O Amazon Q está profundamente integrado ao AWS Cloud, o que permite aos programadores buscar informações detalhadas sobre arquitetura de sistemas, configuração de serviços e otimização de aplicações baseadas na nuvem. Além de auxiliar na escrita de código, a

ferramenta também contribui para a automação de tarefas e a melhoria do desempenho dos serviços em nuvem, facilitando a adoção de boas práticas no ambiente da AWS.

Tabnine (<https://www.tabnine.com>): É uma ferramenta de IA generativa focada em aumentar a produtividade dos desenvolvedores por meio de sugestões de código. Uma característica importante do Tabnine é sua integração com editores de código como Visual Studio Code, IntelliJ e outros, oferecendo suporte a diversas linguagens de programação. Ele também é configurável, permitindo que os desenvolvedores ajustem a ferramenta para suas preferências específicas. O Tabnine pode ser visto como uma alternativa ao GitHub Copilot, pois ambos são assistentes de programação baseados em IA generativa que oferecem sugestões de código. A principal diferença está na privacidade e execução, já que o Tabnine permite processamento local, enquanto o Copilot opera apenas na nuvem.

Gemini (<https://gemini.google.com>): Desenvolvido pelo Google DeepMind, é uma família de modelos de IA generativa que pode ser usada para auxiliar programadores em diversas tarefas. Ele é capaz de gerar, corrigir e explicar trechos de código em várias linguagens de programação, além de ajudar a entender erros e sugerir otimizações. O Gemini está integrado ao Google Colab e ao Android Studio, facilitando

o uso para desenvolvedores que trabalham com Python e desenvolvimento mobile. Além disso, por meio do Google AI Studio, os programadores podem interagir com a IA para explorar soluções e melhorar a qualidade do código.

Claude (<https://claude.ai>): Desenvolvido pela Anthropic, é uma IA generativa que pode auxiliar programadores na escrita, revisão e depuração de código. Embora não seja especializado exclusivamente em programação, o Claude pode interpretar descrições de problemas, sugerir soluções em diversas linguagens e explicar conceitos técnicos de forma clara e acessível. Sua abordagem focada na segurança e controle do usuário faz com que seja uma opção interessante para desenvolvedores que desejam um assistente confiável para tarefas de programação. Ele pode ser utilizado diretamente em navegadores e integrado a fluxos de trabalho por meio de APIs, sendo útil tanto para iniciantes quanto para profissionais experientes.

Considerações finais sobre as ferramentas

O campo das IAs generativas para programação está em constante evolução, com novas ferramentas surgindo regularmente e outras sendo descontinuadas ou substituídas por versões mais avançadas. No momento em que este livro foi escrito e publicado, apresentamos

algumas das principais tecnologias disponíveis, mas é possível que, ao longo do tempo, algumas delas tenham sido aprimoradas, renomeadas ou até deixado de existir. Da mesma forma, novas soluções inovadoras podem surgir, oferecendo funcionalidades ainda mais avançadas. Por isso, incentivamos os leitores a sempre acompanharem as novidades do setor e explorarem as ferramentas mais atuais para obter o máximo benefício do uso da IA na programação.

7 O uso de IAs generativas em empresas de desenvolvimento de software

Uma pesquisa realizada em 2024 com mais de 15 mil programadores brasileiros revelou que a adoção de inteligência artificial no desenvolvimento de software é uma realidade consolidada, com 83,61% dos participantes dizendo utilizar ferramentas de IA em suas atividades profissionais (CÓDIGO FONTE TV, 2024). A mesma pesquisa encontrou que em torno de 88% dos desenvolvedores sentem-se mais produtivos usando IAs, totalmente (62,69%) ou parcialmente (25,65%). Esses dados refletem a crescente integração dessas tecnologias no dia a dia dos programadores, auxiliando em tarefas como geração de código, depuração de erros e automação de processos.

Com a capacidade de interpretar comandos em linguagem natural e gerar soluções rapidamente, as IAs podem auxiliar desenvolvedores na escrita de código mais eficiente, na depuração de erros e até na sugestão de melhorias na arquitetura do sistema. Outra pesquisa mostrou que o GitHub Copilot tem se destacado como uma ferramenta para aumentar a produtividade dos desenvolvedores. A pesquisa descobriu que o Github Copilot já é responsável por gerar até 46% do código e que contribui para uma aceleração de até 55% no processo de codificação (DOHMKE,

2024). Esse impacto se deve à capacidade do Copilot de sugerir trechos de código, completar funções e até antecipar soluções com base no contexto do projeto, reduzindo significativamente o tempo gasto em tarefas repetitivas. A mesma pesquisa encontrou que em torno de 80% dos programadores utilizam o ChatGPT (66,65%) e o GitHub Copilot (13,51%).

Cuidados e preocupações

Apesar dos benefícios evidentes, o uso dessas ferramentas exige um olhar crítico por parte dos programadores, uma vez que as sugestões geradas podem conter erros ou não seguir as melhores práticas, tornando essencial a validação humana para garantir a qualidade e a segurança do código. Erros causados por alucinações podem comprometer a qualidade do software. Outra preocupação importante está na segurança e privacidade dos dados, já que muitas dessas ferramentas operam na nuvem e podem armazenar ou processar informações sensíveis, como dados de clientes e empresas. Os desenvolvedores de software precisam avaliar se o uso dessas IAs pode expor código proprietário ou segredos industriais a terceiros, além de verificar a conformidade com regulamentações como a LGPD (Lei Geral de Proteção de Dados).

Questões éticas e legais também são uma preocupação, pois muitas IAs generativas são treinadas com códigos de diversas fontes, incluindo

repositórios públicos. Isso pode gerar incertezas sobre direitos autorais e licenciamento, especialmente se fragmentos de código protegidos por copyright forem sugeridos sem a devida referência aos autores. As empresas precisam garantir que o uso dessas ferramentas esteja alinhado às políticas de conformidade e evitar violações de propriedade intelectual.

Há também o desafio da cultura organizacional e a adaptação das equipes. A introdução de ferramentas de IA pode mudar significativamente a dinâmica de trabalho, exigindo que os desenvolvedores aprendam a interagir com essas tecnologias de forma eficiente. Algumas empresas podem enfrentar resistência à adoção dessas ferramentas, seja por preocupações dos colaboradores com substituição de empregos ou pela necessidade de requalificação dos profissionais. Para garantir uma transição bem-sucedida, é essencial investir em treinamento, diretrizes claras de uso e integração gradual da IA no fluxo de desenvolvimento de software.

Programação em Pares

Outra discussão interessante está relacionada com a adoção por muitas empresas à prática de programação em pares (pair programming). Na programação em pares, há o envolvimento de dois programadores: o piloto, que escreve código, e o navegador, que revisa o código (BIRD et

al., 2023; WILLIAMS; KESSLER, 2002). A prática da programação em pares está relacionada à metodologia Extreme Programming (XP), sendo uma de suas principais técnicas para melhorar a qualidade do código e a colaboração entre os desenvolvedores. Na metodologia XP, dois programadores trabalham juntos no mesmo código, onde um escreve enquanto o outro revisa em tempo real, garantindo a detecção precoce de erros e a troca contínua de conhecimento. Essa abordagem contribui para a produção de software mais confiável e sustentável, além de fortalecer o trabalho em equipe e a adaptação rápida a mudanças.

No entanto, a aplicação da programação em pares enfrenta desafios como a resistência dos desenvolvedores à colaboração intensa, especialmente em equipes acostumadas ao trabalho individual. Além disso, a prática pode gerar aumento no tempo de desenvolvimento inicial, pois exige comunicação constante e alinhamento entre os pares. Diferenças de nível de experiência entre os programadores também podem impactar a produtividade, tornando necessário um equilíbrio para que ambos contribuam de forma significativa. Outro desafio é a fadiga mental, já que a concentração prolongada e a necessidade de argumentação contínua podem ser exaustivas. Apesar das dificuldades, quando bem implementada, a programação em pares pode resultar em código de maior qualidade e na melhoria do aprendizado da equipe.

As IAs generativas, como o ChatGPT, podem ser utilizadas como parceiras na programação em pares, auxiliando desenvolvedores na

escrita de código, na depuração de erros e na sugestão de melhorias (MA et al., 2024). Esse cenário dá origem ao conceito de programação em pares IA-humano, onde a inteligência artificial atua como um colaborador virtual, oferecendo insights em tempo real e acelerando o processo de desenvolvimento (MA; WU; KOEDINGER, 2023b). Embora a IA não substitua o pensamento crítico e a criatividade humana, sua integração pode otimizar a produtividade nas empresas e ampliar as possibilidades na resolução de problemas complexos.

Por fim, a programação em pares pode ser transportada para o ensino de programação, através do conceito de programação em pares IA-aluno, onde a inteligência artificial atua como um parceiro para estudantes em seu processo de aprendizado. O IA pode fornecer sugestões de código, explicar conceitos, corrigir erros e até propor desafios personalizados, permitindo que os alunos pratiquem e aprimorem suas habilidades de forma mais autônoma e eficiente.

8 O uso de IAs generativas na aprendizagem

Algumas estratégias de ensino apoiado por IAs generativas com benefícios baseados em evidências podem ser integradas por professores em suas aulas, independente da área, como programação, matemática, história ou sociologia. Os alunos poderiam aprender estas estratégias e aplicá-las em seus estudos (MOLLICK; MOLLICK, 2023).

Uma das estratégias é usar a IA para produzir exemplos variados de soluções para um mesmo problema (KIRSCHNER et al., 2022). Quando confrontados com conceitos novos e complexos, ter à disposição exemplos variados sobre os mesmos conceitos pode ajudar os alunos a entendê-los melhor. No entanto, produzir muitos exemplos de um mesmo conceito pode ser demorado para um professor. O próprio aluno poderia pedir à IA generativa exemplos variados para um conceito ou problema (MOLLICK; MOLLICK, 2023).

Os primeiros trabalhos que estudaram o uso das LLMs (modelo de IA generativa abordado no capítulo 3) na educação foram mais focados nas suas capacidades. Os primeiros estudos verificaram se essas IAs poderiam performar bem em provas e solucionar problemas de programação (CIPRIANO; ALVES, 2023; FINNIE-ANSLEY et al.,

2022, 2023; LEINONEN et al., 2023; REEVES et al., 2023). Também havia uma preocupação quanto ao aumento de fraudes em trabalhos e provas e os efeitos disso para os níveis de aprendizagem alcançados (MALINKA et al., 2023).

Pesquisas ainda mais recentes propuseram abordagens para ajudar os alunos a melhorar suas habilidades em programação. Um dos trabalhos (DENNY et al., 2024b) apresentou aos alunos exercícios de programação em formatos visual ou textual. O aluno então precisava escrever *prompts* que pedem à IA uma solução de código para aquele problema. O código gerado pela IA era usado de entrada para uma ferramenta de testes. O objetivo foi verificar se os alunos conseguiriam escrever bons *prompts*, no sentido de fazer a IA compreender suas intenções e gerar uma solução correta para o problema. Os pesquisadores encontraram que a maioria dos alunos que participaram da pesquisa conseguiram escrever bons prompts e resolveram as tarefas em algumas poucas tentativas, embora alguns estudantes tenham precisado de 20 tentativas ou mais. Em geral, os alunos relataram experiências muito positivas e indicaram que a interação com a IA através dos prompts expôs a eles novos conceitos de programação.

Outro trabalho (MACNEIL et al., 2023) criou um *e-book* sobre desenvolvimento web que possui botões próximos de trechos de códigos que, quando acionados, exibem diferentes tipos de explicações sobre os códigos, geradas por uma LLM.

Outro trabalho (JURY et al., 2024) descreve uma ferramenta que automaticamente gera exemplos trabalhados para os estudantes aprenderem programação. Exemplo trabalhado (do inglês worked example) é uma técnica pedagógica utilizada especialmente em áreas como matemática, ciências e programação. Consiste em apresentar para o aluno uma solução completa para um problema ou exercício, com cada passo do processo de resolução sendo explicitamente mostrado e explicado. Essa abordagem facilita a aprendizagem ao reduzir a carga cognitiva dos estudantes (ATKINSON et al., 2000).

O trabalho de Ma et al. (2024) apresentou um sistema que ajuda o aluno a desenvolver uma importante habilidade em programação, a de análise e depuração de códigos. O sistema usa a IA para gerar códigos com erros e o aluno precisa analisá-los e criar hipóteses sobre a causa dos erros.

O uso de IAs generativas no ensino de programação é um campo de estudo recente, com os primeiros artigos publicados em 2023, e que ainda se encontra em estágio inicial de investigação acadêmica. Embora essas ferramentas tenham demonstrado potencial para facilitar o aprendizado, auxiliando na geração de código, explicação de conceitos e depuração de erros, há muitas lacunas de pesquisa a serem exploradas. Questões como o impacto real dessas ferramentas na aprendizagem, a dependência excessiva dos alunos, e a eficácia em diferentes perfis de estudantes ainda não foram totalmente compreendidas.

Além disso, é necessário investigar como adaptar abordagens pedagógicas para integrar essas tecnologias de forma eficaz, garantindo que os alunos desenvolvam pensamento computacional e habilidades de resolução de problemas sem apenas depender da IA para obter respostas prontas. Estudos futuros podem explorar estratégias para ensinar os alunos a usar a IA de maneira crítica e produtiva, além de avaliar o papel do professor na mediação desse processo. Como as IAs generativas estão em constante evolução, é fundamental acompanhar suas implicações no ensino a longo prazo, garantindo que sejam utilizadas como aliadas do aprendizado e não como substitutas do esforço cognitivo necessário para se tornar um programador competente.

9 Riscos de usar IAs generativas na aprendizagem de programação

O uso de IAs generativas na aprendizagem de programação, embora ofereça vantagens significativas, também apresenta riscos que precisam ser cuidadosamente considerados.

Estas ferramentas têm gerado uma preocupação crescente entre professores em relação ao plágio de trabalhos acadêmicos. Com a facilidade de obter respostas prontas e textos gerados automaticamente, muitos alunos podem ser tentados a utilizar essas ferramentas para apresentar trabalhos como se fossem de sua autoria, sem realizar o devido esforço intelectual. Esse comportamento não só compromete a integridade acadêmica, mas também prejudica o processo de aprendizado, pois o aluno deixa de desenvolver habilidades essenciais de pesquisa, análise crítica e expressão própria. Para os professores, é desafiador identificar e lidar com o plágio em um cenário onde a IA pode gerar textos complexos e convincentes, tornando necessária a adoção de novas estratégias de avaliação - provas e demais instrumentos avaliativos -, que incentivem a originalidade e a reflexão, além de ferramentas tecnológicas para detectar a utilização indevida dessas ferramentas (BECKER et al., 2023).

Outro risco importante no uso de ferramentas como o GitHub Copilot é a dependência excessiva dos alunos, especialmente quando essas ferramentas estão integradas diretamente nas IDEs, como o Visual Studio Code e o IntelliJ. Ao oferecer sugestões automáticas enquanto o código é escrito, o Copilot pode facilitar o desenvolvimento, mas também pode fazer com que os alunos se tornem menos críticos em relação ao seu próprio processo de aprendizagem. A constante dependência dessas sugestões pode reduzir a habilidade do aluno de resolver problemas por conta própria e de desenvolver um raciocínio lógico e criativo, fundamentais para a programação. Além disso, ao confiar excessivamente nas respostas automáticas, o aluno pode deixar de compreender profundamente os conceitos envolvidos no código gerado, prejudicando seu aprendizado a longo prazo e a capacidade de enfrentar desafios mais complexos no futuro (BECKER et al., 2023).

Ao depender dessas ferramentas para gerar código, exemplos ou explicações, os alunos também podem acabar absorvendo soluções inadequadas ou imprecisas, sem a devida compreensão dos conceitos subjacentes. A confiança excessiva nas sugestões da IA pode diminuir a capacidade do aluno de desenvolver habilidades de resolução de problemas e raciocínio lógico, essenciais para a programação. Os erros ou limitações das IAs, como alucinações ou respostas imprecisas, discutidas anteriormente neste livro, também podem levar à formação de conceitos errôneos. Portanto, é fundamental que as IAs sejam usadas

como uma ferramenta complementar, e não como substituto do processo ativo de aprendizagem.

10 Estratégias para o aluno se beneficiar das IAs generativas na aprendizagem de programação

Após ler o capítulo anterior sobre os riscos de usar ferramentas de IA generativa para a aprendizagem de programação, o leitor pode se questionar: **Como, então, o aluno pode se beneficiar dessas tecnologias?** Algumas estratégias já foram apresentadas no capítulo 8, mas lá foram trazidas dentro de um contexto geral de aprendizagem, não especificamente sobre a aprendizagem em programação de computadores. Um exemplo apresentado lá é o próprio aluno pedir à IA generativa por exemplos variados para um determinado conceito. Uma solicitação como essa de um aluno para um professor poderia ter uma devolutiva demorada pelo professor, que precisa atender turmas com 10, 20, 30 ou até mesmo 40 alunos. Ao pedir estes exemplos à IA, o aluno teria um retorno muito mais rápido (MOLLICK; MOLLICK, 2023).

Neste capítulo apresentamos **10 estratégias** que poderiam ser adotadas por alunos ou serem propostas por professores durante atividades de aprendizagem de programação em cursos presenciais ou à distância. As estratégias foram desenvolvidas a partir de discussões e reflexões dos autores deste livro, que possuem ampla experiência no

ensino de programação e/ou em estudos de Informática aplicada à Educação. Embora a eficácia e o impacto dessas estratégias na melhoria do aprendizado dos alunos não tenham sido verificadas pelos autores através de métodos científicos, quantitativa ou qualitativamente, acreditamos que elas podem contribuir significativamente para o processo de ensino, com base na experiência dos autores em aplicar estas estratégias com turmas de alunos e nos feedbacks obtidos. As estratégias foram todas aplicadas pelos autores com turmas de alunos, na maior parte para turmas de disciplinas introdutórias de programação e algoritmos. As observações realizadas pelos autores são descritas junto às estratégias, mais adiante neste livro.

Achamos importante destacar que, no momento da escrita e da publicação deste livro, diversas pesquisas na área de Informática Aplicada à Educação estão em andamento e devem divulgar seus resultados científicos durante os próximos anos. Apesar disso, os autores consideram importante compartilhar os insights gerados por suas experiências práticas com alunos, com o intuito de oferecer uma visão pragmática sobre o tema.

Nós, autores, ficaríamos muito felizes em receber feedbacks dos leitores, especialmente daqueles que estão aprendendo a programar e colocaram em prática as estratégias apresentadas. Também gostaríamos de receber feedbacks dos professores que aplicaram as estratégias em aula ou incentivaram a adoção delas pelos alunos. Saber como essas

abordagens foram aplicadas no dia a dia dos estudantes, quais desafios eles enfrentaram e quais benefícios perceberam é fundamental para enriquecer a discussão sobre o uso de IAs generativas na aprendizagem de programação. Além disso, esses relatos podem contribuir para aprimorar futuras edições do livro e inspirar novas pesquisas na área de Informática Aplicada à Educação. Assim, incentivamos os leitores a compartilharem suas experiências, reflexões e sugestões, pois acreditamos que a construção do conhecimento se fortalece com a troca entre educadores e aprendizes.

É fundamental observar que os autores buscaram alinhar as estratégias com uma perspectiva construtivista de aprendizagem, na qual os alunos constroemativamente o conhecimento por meio de experiências e interações, em vez de receberem informações de forma passiva (BECKER et al., 2023). Nesse sentido, as estratégias apresentadas a seguir adquirem significado pleno somente quando os leitores as experimentam, adaptam e refletem sobre sua aplicação prática no aprendizado de programação. Por essa razão, os autores valorizam os feedbacks dos leitores, pois esses relatos enriquecem não apenas a compreensão de como as estratégias se manifestam em diferentes contextos, mas também promovem uma troca de conhecimento mais robusta. Assim, a construção do saber torna-se um processo colaborativo e contínuo, em que alunos e professores aprendem e evoluem em conjunto.

Estratégia 1

Após o aluno codificar com sucesso uma solução, ele pode solicitar à IA que exiba exemplos de soluções de código alternativas à sua.

A primeira estratégia proposta é o aluno **solicitar exemplos de soluções alternativas à sua**. Essa estratégia poderia possibilitar ao aluno explorar diferentes abordagens para resolver o mesmo problema, ampliando sua compreensão sobre lógica de programação, estruturas de dados e padrões de código. Ao comparar sua própria implementação com as alternativas sugeridas pela IA, o aluno pode identificar vantagens e desvantagens de cada abordagem, além de aprender novas técnicas que talvez não tivesse considerado. Essa prática poderia estimular o pensamento crítico na resolução de problemas, habilidades essenciais para se tornar um programador mais eficiente e adaptável aos diferentes problemas apresentados a ele.

Quando aplicamos esta estratégia com nossos alunos, notamos que muitos deles avaliaram de forma crítica o que criaram, questionando a eficiência e a elegância do seu código. Alguns alunos ficaram surpresos ao perceberem que existem formas mais rápidas ou mais simples de resolver um determinado exercício de programação, o que poderia

aumentar a curiosidade e o desejo deles de explorar diferentes técnicas de codificação.

Achamos que essa estratégia pode contribuir para o desenvolvimento de habilidades de resolução de problemas e de pensamento lógico, já que os alunos precisam justificar por que uma solução alternativa pode ser mais adequada em determinadas circunstâncias. Vemos também que, ao comparar soluções, eles começam a expandir seu repertório em programação, o que os torna mais flexíveis e capazes de adaptar suas abordagens a novos desafios.

Estratégia 2

O aluno pode solicitar à IA que esclareça os conceitos presentes em seu código, bem como outros fundamentos de programação que esteja estudando.

Após escrever uma solução de código para um exercício, o aluno poderia interagir com a IA, solicitando uma explicação detalhada sobre o funcionamento de cada linha. Esse tipo de explicação poderia ajudar o aluno a reforçar sua compreensão teórica, além de melhorar a capacidade de comunicar e justificar as suas decisões tomadas. Além disso, a IA pode fornecer uma visão mais ampla sobre os conceitos relacionados, como estruturas de dados, algoritmos ou paradigmas de programação, conectando as partes do código a princípios fundamentais de engenharia de software e lógica computacional.

Isso é especialmente útil para esclarecer dúvidas em tempo real, evitando que o aluno apenas escreva o código sem compreender profundamente o que está fazendo. É muito comum observar alunos que escrevem seus códigos sem entender plenamente o que está acontecendo ali. Ao adotar essa estratégia, o aluno tem a oportunidade de aprofundar

sua formação teórica enquanto aplica os conceitos, promovendo um aprendizado mais sólido e duradouro.

Quando aplicamos essa estratégia com os alunos, notamos que ela realmente pode promover um entendimento mais aprofundado sobre os conceitos. Observamos que muitos alunos inicialmente escrevem o código de forma mecânica, muitas vezes copiando códigos de exercícios parecidos. Ao pedir explicações para a IA, começamos a perceber um engajamento mais reflexivo. Alguns alunos passaram a questionar o "porquê" por trás de cada escolha de código, o que os ajuda a fortalecer sua compreensão dos conceitos fundamentais de programação, como estruturas de controle, fluxos de dados e algoritmos.

Estratégia 3

O aluno pode solicitar à IA que o auxilie na interpretação das mensagens de erro.

Mensagens de erros são uma parte natural do processo de programação, não é mesmo? E muitas vezes as mensagens fornecidas pelas linguagens de programação podem ser vagas ou difíceis de serem interpretadas, especialmente para iniciantes. Ao pedir para a IA explicar o significado da mensagem de erro e sugerir possíveis causas, o aluno pode obter uma compreensão mais clara do que está acontecendo no código e como corrigir o problema. A IA pode, ainda, fornecer exemplos de código corrigido ou explicar o que precisa ser alterado para que o programa funcione corretamente.

Essa estratégia poderia permitir ao aluno não apenas corrigir o erro imediato, mas também aprender sobre os conceitos subjacentes que causaram o problema, como tipos de dados incompatíveis, variáveis não inicializadas ou erros de sintaxe. Ao entender os erros e aprender a corrigi-los, o aluno poderia desenvolver uma mentalidade de resolução de problemas e melhorar sua habilidade de depuração de código. Essa prática também pode aumentar a confiança do aluno, já que muitas vezes um aluno iniciante em programação pode ficar muito tempo trancado

numa mensagem de erro, principalmente quando está praticando programação em casa, sem o suporte do professor. Essa estratégia também pode fortalecer a habilidade de depuração de código.

Nas aulas de programação, percebemos que muitos estudantes se sentem frustrados com as mensagens de erro. Linguagens de programação como o Python realmente emitem mensagens de erro muitas vezes difíceis de interpretar. Percebemos que pedir ajuda à IA parece aliviar essa frustração. A IA oferece uma explicação clara e direcionada, o que permite aos alunos identificar rapidamente o que deu errado e como corrigir. Essa interação não só ajuda a resolver o problema imediato, mas também poderia promover a aprendizagem ativa, pois percebemos que muitos alunos começaram a entender o significado das mensagens de erro e a aplicar correções.

No entanto, notamos que ainda há a necessidade de orientar os alunos para quando pedirem ajuda sobre as mensagens de erro, de não pedirem a solução completa para o problema. Afinal, queremos garantir que as respostas da IA não substituem o pensamento crítico necessário para o estudante entender o problema.

Estratégia 4

O aluno pode consultar a IA para verificar se sua solução de código está correta, deixando explícito no prompt que não deseja receber uma solução final caso esteja incorreto.

Nesta estratégia, após finalizar a codificação de um exercício, o aluno poderia copiar sua solução de código e enviar para a IA juntamente de um prompt como o seguinte:

Veja o código abaixo para o seguinte exercício: “Implemente um algoritmo que recebe dois valores inteiros da entrada do teclado e exibe na tela o produto desses valores.”. Responda apenas com “Solução correta” ou “Solução incorreta”, sem apresentar uma solução correta caso meu código esteja incorreto.

Essa abordagem poderia permitir que o aluno mantenha a autonomia de encontrar a resposta por conta própria, ao mesmo tempo que recebe um feedback mínimo (“Correto” ou “Incorreto”) sobre a correção da solução. Poderia também tornar o aluno mais confiante

sobre suas habilidades de codificação, já que a IA se torna uma ferramenta de validação, em vez de uma fonte para respostas prontas.

Ao observar os alunos aplicando essa estratégia, notamos que eles sentem mais confiança sobre suas soluções estarem realmente corretas, pois a IA apenas responde com um feedback mínimo (“Solução Correta” ou “Solução Incorreta”). É comum ocorrer de um aluno finalizar um exercício de programação acreditando que desenvolveu uma solução correta e completa para um problema, quando, na verdade, a solução atende corretamente apenas parte das entradas possíveis do algoritmo.

Percebemos também que muitos alunos se sentem estimulados a corrigir os códigos por si mesmos, a partir do feedback mínimo apresentado pela IA. Isso parece dar aos alunos uma sensação de realização ao resolver problemas por conta própria. Em alguns casos, observamos que, após o feedback, os alunos não apenas corrigiam os erros, mas também faziam ajustes significativos em suas abordagens, explorando alternativas e otimizando seu código. Esse tipo de independência é crucial para o desenvolvimento de habilidades de programação, e acreditamos que, ao reforçar esse comportamento, estamos ajudando os alunos a se tornarem mais auto suficientes e preparados para enfrentar desafios mais complexos no futuro.

Por fim, esta estratégia agiliza a dinâmica das aulas, pois o aluno não precisa ficar esperando o professor visualizar seu código para dar o chamado feedback mínimo (“Correto” ou “Errado”).

Estratégia 5

O aluno pode solicitar à IA que recomende exercícios.

Uma estratégia interessante é o aluno pedir para a IA sugerir exercícios de programação. Isso permite ao aluno personalizar sua prática, ajustando os exercícios ao seu nível de conhecimento e aos conceitos que ele está estudando no momento. A IA pode gerar desde problemas mais simples, para reforçar conceitos básicos, até desafios mais avançados, que incentivem o aluno a aprofundar seus conhecimentos e a desenvolver novas habilidades.

A estratégia também atende alunos que costumam pedir por mais exercícios ao professor, mas o professor não teve tempo para elaborá-los e disponibilizá-los.

Exemplo de prompt:

“Gere 10 enunciados de exercícios de programação em JavaScript envolvendo arrays e a estrutura de repetição FOR, desde exercícios bem simples até alguns com desafio médio. Deixe explícito no enunciado quais as entradas e as saídas esperadas.”

Ao observar os alunos aplicando esta estratégia, notamos um aumento no engajamento de alguns estudantes, por terem acesso a uma fonte de desafios personalizados. Percebemos que alguns alunos, ao conseguir identificar lacunas em seu conhecimento, pediam por exercícios envolvendo aqueles conhecimentos que consideram mais deficitários. Percebemos também que é importante orientar os alunos sobre pedir exercícios mais adequados ao seu nível, para que não se sintam sobrecarregados por problemas muito difíceis ou desmotivados com exercícios muito fáceis.

Estratégia 6

O aluno pode solicitar à IA a tradução de códigos de uma linguagem de programação para outra.

Outra estratégia que consideramos interessante é o aluno pedir para que a IA traduza códigos implementados em uma linguagem de programação para outra. O objetivo é o estudante conseguir identificar as diferenças entre as linguagens e compreender como os mesmos conceitos e estruturas podem ser expressos de maneiras distintas. Ao fazer isso, o aluno não apenas aprende as particularidades sintáticas de cada linguagem, mas também pode desenvolver uma compreensão mais profunda dos conceitos subjacentes, como variáveis, estruturas de controle, manipulação de dados e funções. Essa prática pode ser muito útil, por exemplo, ao comparar linguagens como Python e Java, que possuem formas distintas de implementar o mesmo algoritmo.

Ao observar as diferenças entre as implementações, o aluno poderia começar a perceber como certos conceitos, como tipagem de dados, gerenciamento de memória e paradigmas de programação, podem ser tratados de maneira diversa em diferentes linguagens. Além disso, ao realizar essa comparação, o aluno fortalece sua capacidade de adaptação

ao aprender novas linguagens de programação e fica mais apto a mudar de contexto ou trabalhar com diferentes tecnologias.

Ao observar os alunos aplicando esta estratégia, percebemos um aumento considerável na compreensão de determinados conceitos, como a tipagem dinâmica. Por exemplo, um aluno que inicialmente tinha dificuldades em entender o conceito de tipagem dinâmica em Python começou a perceber, ao comparar com Java, como a tipagem influencia a estrutura do código e o controle de erros. Essa estratégia parece também ter ajudado a desenvolver uma visão mais crítica sobre as ferramentas que eles têm à disposição, permitindo que tomem decisões mais informadas sobre qual linguagem usar em diferentes contextos.

Também notamos que, ao praticar essa estratégia, os alunos começaram a demonstrar mais flexibilidade cognitiva. Eles passaram a se sentir mais confortáveis em mudar entre diferentes linguagens e paradigmas nas diferentes disciplinas do curso, o que é uma habilidade essencial para desenvolvedores modernos. Um aluno que inicialmente se mostrava relutante em aprender novas linguagens agora estava mais disposto a experimentar e entender as vantagens e limitações de cada uma delas.

Estratégia 7

O aluno pode solicitar à IA sugestões para aprimorar seu código, como a indicação de padrões de projeto e melhores práticas.

Uma estratégia valiosa que um aluno pode adotar em seus estudos de programação é pedir à IA por dicas para melhorar seu código, como sugestões de padrões de projeto ou melhores práticas de codificação. Ao fazer isso, o aluno tem a oportunidade de aprimorar não só a eficiência e legibilidade do seu código, mas também de se aproximar de uma programação mais profissional e bem estruturada. A IA pode ajudar o aluno a identificar áreas onde o código pode ser refatorado para ser mais claro, eficiente e manutenível, e também pode sugerir a aplicação de boas práticas como documentação adequada, nomes significativos para variáveis e funções e testes automatizados.

Com o tempo, espera-se que o aluno passe a internalizar essas práticas e a implementá-las naturalmente em seu trabalho, o que resulta em um código de maior qualidade e maior potencial para ser compreendido por outros desenvolvedores. Essa estratégia também pode preparar o aluno para atuar em ambientes profissionais, onde padrões de

projeto e boas práticas são essenciais para o desenvolvimento de software de alta qualidade.

Ao observar os alunos aplicando a estratégia, notamos uma evolução na qualidade dos códigos deles. Eles também se mostraram mais dispostos a adotar melhores práticas, como nomeação adequada de variáveis e a inclusão de comentários explicativos. Isso nos faz acreditar que a IA não só estaria ajudando os alunos a melhorar tecnicamente, mas também estaria os preparando melhor para atuar com maior profissionalismo no desenvolvimento de software.

Estratégia 8

O aluno pode solicitar à IA para que forneça exemplos práticos sobre como implementar determinados conceitos de programação (worked examples).

Outra estratégia que um aluno poderia aplicar nos seus estudos de programação é pedir que a IA forneça exemplos práticos de como implementar determinados conceitos de programação. Esses exemplos, também conhecidos como worked examples, são extremamente valiosos para o aluno, pois permitem que ele visualize como um conceito teórico é aplicado em código real. Worked examples são exemplos resolvidos de problemas que demonstram, passo a passo, como aplicar conceitos ou métodos para chegar a uma solução. Essa abordagem detalhada ajuda os alunos a compreender o raciocínio e os procedimentos necessários, facilitando a aprendizagem de novos conteúdos e técnicas. Por exemplo, se o aluno está estudando estruturas de dados como listas ou árvores binárias, ele pode solicitar à IA um exemplo de código que mostre como essas estruturas podem ser implementadas e manipuladas. Isso ajuda o aluno a entender não apenas o que o conceito significa, mas também

como ele se traduz em código, fornecendo uma base prática para a aplicação do conceito.

Ao observar os alunos aplicando esta estratégia, percebemos uma melhora na compreensão de conceitos. Vimos que essa estratégia ajudou especialmente aqueles alunos que tinham dificuldade em "ver" o conceito funcionando. Com o exemplo concreto, eles passaram a perceber como os elementos do código interagem e a entender o fluxo de execução. Percebemos também um aumento na confiança dos alunos. Ao verem a implementação de um conceito de forma prática e comentada, muitos alunos começaram a experimentar mais com seus próprios códigos, adaptando os exemplos da IA para suas necessidades.

Estratégia 9

O aluno pode solicitar à IA que elabore uma documentação para o seu código.

O aluno pode pedir à IA para documentar o código implementado por ele. A documentação de código é uma prática essencial para garantir que o código seja facilmente compreendido e mantido, tanto por outros desenvolvedores quanto pelo próprio programador no futuro. Ao solicitar à IA que forneça comentários explicativos, descrições das funções, variáveis e lógica do código, o aluno pode não apenas melhorar a legibilidade do código, mas também aprender a articular o funcionamento de suas soluções de forma clara e comprehensível. As IAs generativas são excelentes para montar documentação e até mesmo tutoriais a partir de códigos fornecidos a ela.

Essa estratégia é especialmente útil para os alunos que estão começando a entender a importância da documentação e muitas vezes não sabem por onde começar. A IA pode fornecer exemplos de como escrever comentários informativos, explicando não apenas o como o código funciona, mas também o porquê de certas escolhas de implementação. Além disso, a IA pode ajudá-lo a identificar partes do

código que podem ser confusas ou difíceis de entender e sugerir melhorias.

Ao observar os alunos aplicando a estratégia, percebemos que essa prática tem ajudado os alunos a desenvolver uma compreensão mais profunda de seu próprio trabalho. Inicialmente, muitos alunos não priorizavam a documentação ou se sentiam inseguros quanto a como comentar adequadamente seu código. Agora, ao solicitarem à IA que fornecesse uma documentação clara e explicativa, muitos alunos passaram a ver a importância dessa prática não apenas como uma exigência acadêmica, mas como uma maneira de refletir sobre suas escolhas de programação.

Também observamos que, ao revisitar o código e os comentários gerados pela IA, os alunos estão refletindo mais criticamente sobre a eficiência e a clareza de suas soluções, o que tem impactado positivamente na qualidade geral de seus projetos entregues.

Estratégia 10

O aluno pode solicitar à IA que recomende estratégias para testar seu código, como a elaboração de planos de testes automatizados.

Esta estratégia parece ser uma excelente forma de os alunos aprimorarem suas habilidades em qualidade de software. Ao solicitar à IA sugestões de testes, o aluno poderia aprender sobre a importância dos testes em diferentes estágios do desenvolvimento de software, como testes unitários, de integração e de aceitação. A IA poderia gerar casos de teste com base nas funcionalidades do código e nas diferentes entradas e saídas esperadas, além de recomendar frameworks e ferramentas de teste automatizado que o aluno pode usar, como JUnit para Java ou PyTest para Python. Ao aplicar essa estratégia, o aluno poderia passar a entender como garantir que seu código funciona corretamente em uma variedade de cenários e se comporta de maneira robusta e confiável. Além disso, o aluno poderia se familiarizar com práticas essenciais de desenvolvimento, como testes contínuos e integração contínua, que são fundamentais no desenvolvimento de software de alta qualidade.

O uso dessa abordagem também poderia ajudar a consolidar o conceito de engajamento proativo com o processo de desenvolvimento e incentivar a prática de desenvolvimento orientado a testes (TDD), que é uma metodologia importante no mundo da programação.

Nós ainda não aplicamos essa estratégia de forma prática com nossos alunos, mas acreditamos que pode ter um potencial interessante. Estamos planejando aplicar essa abordagem em nossas próximas disciplinas, pois vemos nelas uma maneira eficaz de incentivar os alunos a se aprofundarem não só no processo de codificação, mas também nas boas práticas de desenvolvimento de software, como a criação de testes automatizados. Acreditamos que pode trazer benefícios, ajudando os alunos a desenvolverem habilidades cruciais para a criação de códigos robustos e confiáveis.

11 Considerações sobre as estratégias

É importante destacar que as estratégias apresentadas neste livro são baseadas exclusivamente na experiência dos autores com o ensino de programação e o uso de IAs generativas nesse contexto. Embora acreditemos que essas abordagens possam ser úteis para os alunos, sua eficácia ainda precisa ser validada por pesquisas científicas rigorosas, que avaliem quantitativa e qualitativamente seu impacto na aprendizagem. Essas investigações devem considerar diferentes perfis de estudantes e contextos educacionais para fornecer uma análise mais precisa. Muitas dessas pesquisas, inclusive, possivelmente estão sendo conduzidas no momento da publicação deste livro, e seus resultados poderão oferecer uma base mais sólida para a adoção e aprimoramento dessas práticas no futuro.

12 O futuro do uso das IAs generativas na aprendizagem de programação

O futuro do uso das IAs generativas na aprendizagem de programação promete transformar profundamente a forma como os alunos abordam o aprendizado e o desenvolvimento de software. À medida que as IAs se tornam mais avançadas, elas poderão oferecer suporte cada vez mais personalizado e dinâmico, adaptando-se ao ritmo e ao nível de habilidade de cada aluno. Em vez de apenas fornecer respostas ou sugestões, as IAs generativas poderão atuar como mentores digitais, guiando os estudantes através de uma série de desafios progressivos, fornecendo feedback em tempo real e até mesmo criando um ambiente de aprendizagem imersivo e interativo. Essa personalização do ensino pode ser particularmente útil para alunos com diferentes estilos de aprendizagem e para aqueles que enfrentam dificuldades em conceitos específicos.

Além disso, a colaboração entre humanos e IAs nas fases de desenvolvimento de software pode incentivar uma abordagem mais criativa e eficiente na resolução de problemas, permitindo que os alunos se concentrem mais em aspectos de alto nível do desenvolvimento, enquanto as IAs cuidam de tarefas repetitivas ou complexas. À medida

que as IAs continuarem a evoluir, espera-se que elas não só se tornem ferramentas ainda mais poderosas para a programação, mas também desempenhem um papel fundamental no apoio à aprendizagem contínua, ajudando os alunos a acompanhar as mudanças constantes no mundo da tecnologia.

13 Mas e agora? As IAs substituirão os programadores? Devo continuar estudando programação?

Uma pergunta frequente de nossos alunos é se as IAs substituirão os programadores. Qual professor de programação não recebeu esta pergunta nos últimos anos? Os alunos ficam curiosos e, por vezes, preocupados com o impacto das novas tecnologias em suas carreiras, especialmente à medida que ferramentas como o GitHub Copilot e o ChatGPT se tornam mais avançadas.

Como professores, entendemos essa preocupação e buscamos explicar que, embora as IAs possam transformar a maneira como programamos, elas não substituem a necessidade de habilidades humanas essenciais, como criatividade, tomada de decisões e compreensão crítica do contexto. Acreditamos que as IAs irão atuar como ferramentas poderosas que ampliam as capacidades dos programadores, ajudando-os a serem mais produtivos e eficientes. Acreditamos que elas permitirão, cada vez mais, que os programadores se concentrem em aspectos mais estratégicos e criativos do desenvolvimento de software. Elas podem automatizar tarefas repetitivas, sugerir soluções alternativas, gerar códigos e até mesmo

verificar erros, mas a criatividade, a resolução de problemas complexos e o entendimento profundo do contexto continuam sendo habilidades exclusivamente humanas.

Além disso, a programação não se resume apenas à codificação. Ela envolve comunicação, colaboração, definição de requisitos, análise do negócio e dos problemas dos clientes, decisões estratégicas e a capacidade de se adaptar a novos problemas e tecnologias. Estes são aspectos que as IAs, por mais avançadas que sejam, ainda não conseguem realizar de forma autônoma e criativa. Portanto, em vez de substituir os programadores, acreditamos que as IAs permitirão que eles se concentrem mais nos aspectos criativos e de alto nível do desenvolvimento de software, deixando que as máquinas cuidem de tarefas mais repetitivas e técnicas. A chave estará em aprender a usar as IAs de maneira eficaz, aproveitando seu potencial sem perder a essência do trabalho humano que impulsiona a inovação.

Estas conversas com os alunos sempre nos levam a refletir sobre como as ferramentas podem ser utilizadas de maneira inteligente para aprimorar as habilidades do programador, em vez de eliminar a profissão.

Referências

ATKINSON, R. K. et al. Learning from Examples: Instructional Principles from the Worked Examples Research. **Review of Educational Research Summer**, v. 70, n. 2, p. 181–214, 2000.

ATMATZIDOU, S.; DEMETRIADIS, S. Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. **Robotics and Autonomous Systems**, v. 75, p. 661–670, jan. 2016.

BECKER, B. A. et al. **Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation**. Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. **Anais...** Em: SIGCSE 2023: THE 54TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. Toronto ON Canada: ACM, 2 mar. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3545945.3569759>>. Acesso em: 22 ago. 2024

BENNEDSEN, J.; CASPERSEN, M. E. Failure Rates in Introductory Programming. 2007.

BENNEDSEN, J.; CASPERSEN, M. E. Failure rates in introductory programming: 12 years later. **ACM Inroads**, v. 10, n. 2, p. 30–36, abr. 2019.

BIRD, C. et al. Taking Flight with Copilot. **Communications of the ACM**, v. 66, n. 6, p. 56–62, jun. 2023.

CHI, M. T.; OHLSSON, S. Complex declarative learning. **The Cambridge handbook of thinking and reasoning**, v. 858, p. 371–399, 2005.

CIPRIANO, B. P.; ALVES, P. GPT-3 vs Object Oriented Programming Assignments: An Experience Report. Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. **Anais...** Em: ITICSE 2023: INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. Turku Finland: ACM, 29 jun. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3587102.3588814>>. Acesso em: 26 ago. 2024

COCKBURN, A.; WILLIAMS, L. The costs and benefits of pair programming, extreme programming and flexible processes in software engineering. **XP2000, Cagliari, Italy**, 2000.

CÓDIGO FONTE TV. Pesquisa Salarial de Programadores Brasileiros 2024. [s.l: s.n.]. Disponível em: <<https://pesquisa.codigofonte.com.br/2024>>. Acesso em: 22 ago. 2024.

DENNY, P. et al. **Chat Overflow: Artificially Intelligent Models for Computing Education - renAIssance or apocAIypse?** Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. **Anais...** Em: ITICSE 2023: INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. Turku Finland: ACM, 29 jun. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3587102.3588773>>. Acesso em: 22 ago. 2024

DENNY, P. et al. Computing Education in the Era of Generative AI. **Communications of the ACM**, v. 67, n. 2, p. 56–67, fev. 2024a.

DENNY, P. et al. **Prompt Problems: A New Programming Exercise for the Generative AI Era.** Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1. **Anais...** Em: SIGCSE 2024: THE 55TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. Portland OR USA: ACM, 7 mar. 2024b. Disponível em: <<https://dl.acm.org/doi/10.1145/3626252.3630909>>. Acesso em: 15 ago. 2024

DEPARTMENT FOR EDUCATION. **National Curriculum.**, 16 jul. 2014. Disponível em: <<https://www.gov.uk/government/collections/national-curriculum>>. Acesso em: 16 ago. 2024

DOHMKE, T. **GitHub Copilot X: The AI-powered developer experience**. GitHub Blog, 21 maio 2024. Disponível em: <<https://github.blog/news-insights/product-news/github-copilot-x-the-ai-powered-developer-experience/>>. Acesso em: 16 ago. 2024

FINNIE-ANSLEY, J. et al. **The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming**. Proceedings of the 24th Australasian Computing Education Conference. Anais... Em: ACE '22: AUSTRALASIAN COMPUTING EDUCATION CONFERENCE. Virtual Event Australia: ACM, 14 fev. 2022. Disponível em: <<https://dl.acm.org/doi/10.1145/3511861.3511863>>. Acesso em: 26 ago. 2024

FINNIE-ANSLEY, J. et al. **My AI Wants to Know if This Will Be on the Exam: Testing OpenAI's Codex on CS2 Programming Exercises**. Proceedings of the 25th Australasian Computing Education Conference. Anais... Em: ACE '23: AUSTRALASIAN COMPUTING EDUCATION CONFERENCE. Melbourne VIC Australia: ACM, 30 jan. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3576123.3576134>>. Acesso em: 26 ago. 2024

GHIMIRE, A.; EDWARDS, J. Coding with AI: How Are Tools Like ChatGPT Being Used by Students in Foundational Programming Courses. Em: OLNEY, A. M. et al. (Eds.). **Artificial Intelligence in Education**. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2024. v. 14830p. 259–267.

JURY, B. et al. **Evaluating LLM-generated Worked Examples in an Introductory Programming Course**. Proceedings of the 26th

Australasian Computing Education Conference. **Anais...** Em: ACE 2024: AUSTRALIAN COMPUTING EDUCATION CONFERENCE. Sydney NSW Australia: ACM, 29 jan. 2024. Disponível em: <<https://dl.acm.org/doi/10.1145/3636243.3636252>>. Acesso em: 26 ago. 2024

KAZIMOGLU, C. et al. A Serious Game for Developing Computational Thinking and Learning Introductory Computer Programming. **Procedia - Social and Behavioral Sciences**, v. 47, p. 1991–1999, 2012.

KIRSCHNER, P. A. et al. **How teaching happens: seminal works in teaching and teacher effectiveness and what they mean in practice**. London New York: Routledge, Taylor & Francis Group, 2022.

LEHMAN, B. et al. Inducing and tracking confusion with contradictions during complex learning. **International Journal of Artificial Intelligence in Education**, v. 22, n. 1–2, p. 85–105, 2013.

LEINONEN, J. et al. **Using Large Language Models to Enhance Programming Error Messages**. Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. **Anais...** Em: SIGCSE 2023: THE 54TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. Toronto ON Canada: ACM, 2 mar. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3545945.3569770>>. Acesso em: 26 ago. 2024

LOKSA, D. et al. Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. 2016.

LOUKIDES, M. What Are ChatGPT and Its Friends? 2023.

MA, Q. et al. How to Teach Programming in the AI Era? Using LLMs as a Teachable Agent for Debugging. Em: OLNEY, A. M. et al. (Eds.). **Artificial Intelligence in Education**. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2024. v. 14829p. 265–279.

MA, Q.; WU, T.; KOEDINGER, K. **Is AI the better programming**

partner? Human-Human Pair Programming vs. Human-AI pAIr Programming. arXiv, , 8 jun. 2023a. Disponível em: <<http://arxiv.org/abs/2306.05153>>. Acesso em: 22 ago. 2024

MA, Q.; WU, T.; KOEDINGER, K. **Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI pAIr Programming.** arXiv, , 9 jun. 2023b. Disponível em: <<http://arxiv.org/abs/2306.05153>>. Acesso em: 10 out. 2024

MACNEIL, S. et al. **Experiences from Using Code Explanations Generated by Large Language Models in a Web Software Development E-Book.** Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1. *Anais...* Em: SIGCSE 2023: THE 54TH ACM TECHNICAL SYMPOSIUM ON COMPUTER SCIENCE EDUCATION. Toronto ON Canada: ACM, 2 mar. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3545945.3569785>>. Acesso em: 26 ago. 2024

MALINKA, K. et al. **On the Educational Impact of ChatGPT: Is Artificial Intelligence Ready to Obtain a University Degree?** Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. *Anais...* Em: ITICSE 2023: INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. Turku Finland: ACM, 29 jun. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3587102.3588827>>. Acesso em: 26 ago. 2024

MINISTRO DA EDUCAÇÃO. **Base Nacional Curricular.** , 17 fev. 2022.

MOLLICK, E. R.; MOLLICK, L. Using AI to Implement Effective Teaching Strategies in Classrooms: Five Strategies, Including Prompts. **SSRN Electronic Journal**, 2023.

NATIONAL SCIENCE FOUNDATION. **CS For All.** , [s.d.]. Disponível em: <https://www.nsf.gov/news/special_reports/csed/csforall.jsp>. Acesso em: 16 ago. 2024

PORTER, L.; ZINGARO, D. **Learn AI-assisted Python programming: with GitHub Copilot and ChatGPT**. Shelter Island, NY: Manning, 2024.

PRATHER, J. et al. **The Robots Are Here: Navigating the Generative AI Revolution in Computing Education**. Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education. *Anais...* Em: ITICSE 2023: INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. Turku Finland: ACM, 22 dez. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3623762.3633499>>. Acesso em: 16 ago. 2024

REEVES, B. et al. **Evaluating the Performance of Code Generation Models for Solving Parsons Problems With Small Prompt Variations**. Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1. *Anais...* Em: ITICSE 2023: INNOVATION AND TECHNOLOGY IN COMPUTER SCIENCE EDUCATION. Turku Finland: ACM, 29 jun. 2023. Disponível em: <<https://dl.acm.org/doi/10.1145/3587102.3588805>>. Acesso em: 26 ago. 2024

SALLEH, N.; MENDES, E.; GRUNDY, J. Empirical studies of pair programming for CS/SE teaching in higher education: A systematic literature review. **IEEE Transactions on Software Engineering**, v. 37, n. 4, p. 509–525, 2010.

TANG, L. et al. **Solving Probability and Statistics Problems by Probabilistic Program Synthesis at Human Level and Predicting Solvability**. Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners' and Doctoral Consortium: 23rd International Conference, AIED 2022, Durham, UK, July 27–31, 2022, Proceedings, Part II. *Anais...* Berlin, Heidelberg: Springer-Verlag, 2022. Disponível em: <https://doi.org/10.1007/978-3-031-11647-6_127>

TSAI, M.-J.; WANG, C.-Y.; HSU, P.-F. Developing the Computer Programming Self-Efficacy Scale for Computer Literacy Education. **Journal of Educational Computing Research**, v. 56, n. 8, p. 1345–1360, jan. 2019.

UMAPATHY, K.; RITZHAUPT, A. D. A Meta-Analysis of Pair-Programming in Computer Programming Courses: Implications for Educational Practice. **ACM Trans. Comput. Educ.**, v. 17, n. 4, ago. 2017.

WALDO, J.; BOUSSARD, S. GPTs and Hallucination. **Communications of the ACM**, v. 68, n. 1, p. 40–45, jan. 2025.

WHITE, J. et al. **A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT**. arXiv, , 21 fev. 2023. Disponível em: <<http://arxiv.org/abs/2302.11382>>. Acesso em: 27 ago. 2024

WILLIAMS, L.; KESSLER, R. **Pair Programming Illuminated**. USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

Aprendendo a Programar com a Ajuda de IAs: Estratégias para Turbinar Seus Estudos

1ª edição



O avanço das IAs generativas está transformando a aprendizagem de programação. Mas como usá-las estratégicamente sem se tornar dependente? Este livro apresenta **10 estratégias práticas** para potencializar os estudos, desde pedir explicações sobre conceitos até melhorar códigos e criar testes automatizados. Além disso, discute desafios como dependência tecnológica, engenharia de prompts e limitações das IAs. Com uma abordagem acessível e fundamentada na experiência dos autores, a obra ajuda estudantes e educadores a integrar a IA no ensino de programação de forma consciente.

E, afinal, as IAs substituirão os programadores?

Descubra nesta leitura!

Tiago Roberto Kautzmann
Fabrício Malta de Oliveira

Série Aprendizagem de Programação