Avaliando resoluções de exercícios introdutórios de programação na era das IAs Generativas: Um Estudo de Caso com o ChatGPT

William Simão de Deus[1], Anderson da Silva Marcolino[2], Gustavo Martins Nunes Avellar[3], Katyeudo Karlos de Sousa Oliveira[3], Ellen Francine Barbosa[3]

[1]Instituto Federal do Paraná (IFPR)
Pinhais – PR – Brasil

[2]Universidade Federal do Paraná (UFPR)
Palotina – PR – Brasil

[3]Instituto de Ciências Matemáticas e de Computação (ICMC)
Universidade de São Paulo (USP)
São Carlos – SP- Brasil

william.deus@ifpr.edu.br, anderson.marcolino@ufpr.br,

{gustavo.avellar,karlos_oliveira}@usp.br, francine@icmc.usp.br

Abstract. *This paper describes a case study on how ChatGPT approaches and solves introductory computer programming problems, identifying elements that demonstrate whether the code was produced by generative Artificial Intelligence (AI) and not by a beginner in programming. To this end, 1.258 lines of code produced by ChatGPT were analyzed. Results provided patterns and practices that are not usually observed by beginners in programming. This study provides an analysis and suggestions considering the practical application for educators.*

Resumo. *Este artigo descreve um estudo de caso sobre como o ChatGPT aborda e resolve problemas de programação de computadores introdutória, identificando elementos que evidenciam que o código foi produzido por uma Inteligência Artificial (IA) generativa e não por um estudante iniciante em programação. Para isso, foram analisadas 1.258 linhas de código produzidos pelo ChatGPT. Os resultados demonstram padrões e práticas que não são comuns por alunos que estão aprendendo a programar. O artigo apresenta uma análise e recomendações considerando as implicações práticas para professores.*

1. Introdução

A Inteligência Artificial (IA) generativa refere-se ao uso de técnicas computacionais capazes de gerar conteúdos aparentemente novos e significativos [Feuerriegel et al. 2024]. Entre as possibilidades, destaca-se a geração de imagens, textos ou outros tipos de dados conforme a demanda de cada usuário [Euchner 2023].

Entre os diversos usos das IAs generativas, estão a assistência na redação de textos [Imran and Almusharraf 2023], a criação de conteúdos criativos como histórias e poemas [Virvou et al. 2023] e até mesmo o suporte em tarefas administrativas, como a

DOI: 10.5753/sbie.2024.242283

elaboração de *e-mails* e relatórios [Taecharungroj 2023]. No cenário educacional, muitas vezes, IAs generativas são utilizadas para auxiliar na resolução de dúvidas acadêmicas, na elaboração de trabalhos escolares e na prática de línguas estrangeiras [Sharples 2023]. Outro campo da educação em que a IA generativa pode ser empregada é no aprendizado de programação de computadores [Bull and Kharrufa 2023]. Ela auxilia na compreensão dos conceitos principais, fornecendo explicações e exemplos práticos conforme a necessidade dos alunos [Biswas 2023].

No entanto, muitos problemas estão surgindo diante do uso de IAs generativas no contexto educacional. Em primeiro lugar, as IAs generativas demonstram excelente desempenho na produção de soluções para exercícios introdutórios de programação [Denny et al. 2024a]. Isso impacta diretamente a estratégia de ensino que muitos professores de programação introdutória adotam: a resolução de problemas, exercícios e atividades avaliativas [Oliveira et al. 2023].

Em segundo lugar, os softwares que analisam a integridade das soluções produzidas por alunos iniciantes em programação são ineficazes para identificar códigos produzidos por uma IA generativa [Biderman and Raff 2022]. Como o resultado produzido por uma IA generativa depende dos dados de treinamento, é possível que sejam geradas diversas soluções válidas para um mesmo exercício. Isso impõe outro problema aos professores: a dificuldade de avaliar a integridade acadêmica das soluções produzidas por seus estudantes [Denny et al. 2024b].

Diante deste cenário, o principal objetivo deste estudo é apoiar os professores durante a avaliação de exercícios de programação introdutória, identificando como uma IA generativa aborda e resolve problemas introdutórios de programação. Para isso foi realizado um estudo de caso com o ChatGPT, a principal IA generativa utilizada por iniciantes em programação [Stack Overflow 2023]. Ao todo, foram analisadas 1.258 linhas de código geradas por essa IA para solucionar 42 exercícios introdutórios. Os resultados demonstram que há diversos elementos que caracterizam o código de uma IA generativa, diferenciando-se do comportamento de um estudante que está aprendendo a programar.

O restante deste trabalho encontra-se organizado da seguinte forma: a Seção 2 apresenta os fundamentos. A Seção 3 detalha a estrutura do estudo de caso. A Seção 4 sumariza os resultados alcançados. A Seção 5 discute o impacto para a rotina de professores. Por fim, a Seção 6 encerra o artigo.

2. Fundamentos

Uma IA generativa é capaz de produzir conteúdo muito similar aos conteúdos desenvolvidos por seres humanos, incluindo textos, imagens, áudios e até mesmo código-fonte para programação de computadores [Feuerriegel et al. 2024, Denny et al. 2024b]. Isso ocorre pela natureza do seu treinamento. Uma IA generativa utiliza um grande conjunto de dados (como texto, imagens e outros arquivos) para criar, conforme a demanda de usuários, novas versões de conteúdo [Euchner 2023].

Diferentes IAs generativas estão disponíveis aos usuários. O ChatGPT é uma das mais populares, tendo alcançado a marca de 100 milhões de usuários com 2 meses após seu lançamento [Liu et al. 2024]. Como explorado por [Euchner 2023], as IAs generativas usam modelos matemáticos que tentam prever qual é o melhor conteúdo a ser gerado com base um *prompt*.

Isso impõem um desafio aos professores de diferentes áreas. Em programação introdutória, por exemplo, é muito comum o uso de uma abordagem incremental, partindo dos conceitos mais básicos (variável, atribuição de valores) até atingir os conceitos mais adiantados (*loops*, métodos, funções...), como mostra a Tabela 1. Com essa dinâmica, é muito habitual que as IA generativas consigam resolver diferentes exercícios de programação introdutória e que isso seja usado por alunos para trapacear em atividades avaliativas [Denny et al. 2024a].

Tabela 1. Conceitos básicos de programação segundo [Tew and Guzdial 2010]

Conceito	Descrição
Fundamentos	Conceitos elementares, como variáveis, memória, atribuição de valores
Operadores Lógicos	Operadores "E" e "OU"
Estrutura Condicional	IF, ELSE, ELSE IF
Laços de repetição	For, While, Do...While
Arrays	Uso de vetores e matrizes
Métodos/Funções	Parâmetro e retorno
Recursão	Uso de métodos/funções recursivas
Básico de POO	Conceitos sobre objetos, métodos e atributos

Obs: Originalmente, os autores definiram 10 itens, mas optou-se por agrupar os itens semelhantes.

O problema se torna central para os educadores, tendo em vista que muitos utilizam atividades, exercícios e outros tipos de mecanismos de avaliação para verificar o desempenho de seus alunos. Mas, até os dias de hoje, ainda é difícil avaliar se uma resolução de exercício foi desenvolvida por um aluno ou por uma IA generativa.

2.1. Trabalhos similares

Recentemente, [Denny et al. 2024b] analisaram o impacto da IA generativa no ensino e aprendizado de computação focando na perspectiva dos alunos e dos professores. Em uma perspectiva muito similar, [Liu et al. 2024] investigaram o uso do ChatGPT para solucionar problemas de programação. Ambos os trabalhos apresentam como o uso da IA generativa é capaz de influenciar o processo de ensino e aprendizado.

No entanto, os estudos de [Denny et al. 2024b] e [Liu et al. 2024] possuem um perfil de investigação distinto do trabalho aqui relatado. O presente estudo foca em compreender como uma IA generativa aborda e resolve exercícios de programação. Assim, busca fornecer uma contribuição direta aos professores no momento de avaliar a resolução de um exercício.

No contexto nacional, [Filho et al. 2023] também analisaram como o ChatGPT aborda exercícios de programação introdutória. A principal diferença entre o trabalho dos autores e o estudo aqui é a linguagem de programação utilizada. Enquanto [Filho et al. 2023] adotaram a linguagem C, o presente estudo foi desenvolvido usando a linguagem Python.

3. Estudo de Caso

Esta investigação foi desenvolvida por meio de um Estudo de Caso. Para isso, as orientações de [Yin 2015] foram seguidas.

3.1. Questão de pesquisa

A primeira etapa de investigação foi elaborar as Questões de Pesquisa (QP):

- *Como uma IA generativa aborda e resolve exercícios de programação introdutória?*
- *Quais elementos identificam que um exercício introdutório de programação foi presumidamente resolvido por uma IA generativa?*

3.2. Coleta de dados

Considerando a natureza das QPs estabelecidas, bem como o escopo da investigação, a plataforma BeeCrowd[1] foi selecionada. Em linhas gerais, essa plataforma foi escolhida para coleta de dados pois possui uma extensa coleção de exercícios de programação. Os exercícios estão organizados em diferentes níveis de complexidade, variando do 1 (mais simples) até o 9 (mais complexo).

Os exercícios do nível 8 e 9 não foram coletados por envolverem domínios mais avançados de programação, indo além do escopo introdutório. A seguir, a amostra foi balanceada para evitar a ocorrência de muitos exercícios de um determinado nível, em detrimento dos demais, o que poderia enviesar a análise. Esse procedimento foi efetuado com base em [Yin 2015] para evitar que os dados coletados fossem oriundos de um determinado nível apenas, inserindo viés amostral. A Tabela 2 sumariza a coleta de dados na plataforma.

Nível de complexidade	1	2	3	4	5	6	7	8	9
Exercícios	125	82	46	39	28	6	6	2	2
Amostragem	6	6	6	6	6	6	6	0	0

Tabela 2. Complexidade dos exercícios de programação que foram utilizados

Para realizar a solução dos exercícios, foi utilizada a linguagem de programação *Python* e a IA generativa ChatGPT. Ambos foram selecionados tendo em vista que o relatório do [Stack Overflow 2023] identificou o *Python* como linguagem de programação mais utilizada por iniciantes em programação e o ChatGPT como sendo a IA generativa mais comum para esse público.

Com base nessas configurações, os exercícios selecionados do BeeCrowd foram enviados para o ChatGPT com o seguinte *prompt*: *escreva um programa em python que realize o seguinte processamento:* . Logo a seguir, era inserido o enunciado do exercício.

3.3. Preparação e Análise dos dados

Os dados gerados pelo ChatGPT incluíam a resposta do exercício, mas também continham comentários e quebras de linhas. Por esse motivo, os dados gerados foram unificados em um corpus textual. Tal corpus foi revisado pelos pesquisadores, classificando cada linha como sendo *comentário, código* ou *quebras de linhas*.

Logo após, cada linha de código foi inspecionada manualmente pelos pesquisadores. O objetivo dessa inspeção foi classificar os tópicos introdutórios de programação

[1]https://judge.beecrowd.com/pt/login

propostos por [Tew and Guzdial 2010]. Por exemplo, o trecho do código abaixo foi classificado como tendo dois conceitos introdutórios: uma estrutura de condição (IF) e um operador lógico (NOT IN):

```
if ano_karl not in entrada_por_ano:
```

3.4. Síntese

Para facilitar a compreensão do percurso metodológico, a Figura 1 apresenta uma síntese das etapas do Estudo de Caso.

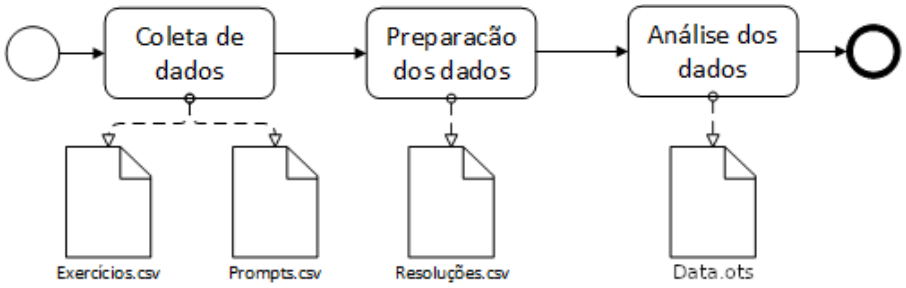

Figura 1. Síntese do Estudo de Caso

Cada etapa produziu diferentes artefatos que estão disponíveis para análise e replicação por outros pesquisadores no repositório: https://www.doi.org/10.5281/zenodo.13312235. Por fim, cabe ainda destacar que foi conduzido um teste piloto utilizando para verificar a adequação das etapas e dos artefatos antes de executar o estudo de caso.

4. Resultados

Ao todo, o ChatGPT gerou 1.258 linhas de código para resolver os 42 exercícios de programação introdutória. No entanto, parte dessas linhas eram de comentários e ou linhas vazias (quebras de linhas) para organização do código. Assim sendo, foram identificadas 839 linhas com código de programação de fato, enquanto as demais linhas eram para comentar e/ou estruturar o código. A seguir, cada QP é detalhada diante da análise realizada nos dados coletados.

4.1. Como uma IA generativa aborda e resolve exercícios de programação introdutória?

Com base nos dados analisados, observa-se que o ChatGPT possui um comportamento avançado de solução de problemas. No geral, a solução é planejada em partes, usando blocos de códigos para organizar a resolução do exercício. Uma análise mais detalhada nos recursos utilizados demonstra diversos comportamentos distintos dos alunos que estão aprendendo a programar. Os principais achados são expostos a seguir.

Variáveis: Ao nomear as variáveis, o ChatGPT deriva informações do próprio enunciado do exercício. Por exemplo, um exercício analisado demonstrava que era necessário *encontrar o alce mais forte no ano*. O ChatGPT criou a variável nomeada como *alce_mais_forte*. Em outro exercício, o enunciado declarava que um número perfeito é a *soma de todos os seus divisores*. O ChatGPT criou a variável *soma_divisores*.

Além disso, os dados analisados evidenciaram que muitas vezes são feitas atribuições por meio de funções, uso encadeado de operadores (como -= ou +=) e por meio de recursos avançados de programação, como exposto abaixo:

```
anos = sorted(entrada_por_ano.keys()) #uso de funcao
denominador = 2 ** (n - 1) #uso de operadores
notas_sem_extremos = notas[1:-1] #uso de recursos avancados
```

Estruturas condicionais (IF, ELSE e ELIF): O uso de estruturas condicionais pelo ChatGPT é muito planejado. A indentação, por exemplo, obedece as normas de programação, mesmo com instruções aninhadas. Ademais, quase sempre são criados blocos de programação separados por quebras de linhas antes e depois do bloco condicional, o que sinaliza facilmente o início e o fim do bloco de instrução, como pode ser visto no código abaixo:

```
[CODIGO]

    if menor_distancia <= 20:
        resultados.append("A")
    elif menor_distancia < 50:
        resultados.append("M")
    else:
        resultados.append("B")

[CODIGO]
```

Ademais, as estruturas de condição são pontos nos quais o ChatGPT otimiza o processamento, evitando escrita de código redundante. Por exemplo, observe abaixo os códigos gerados pelo ChatGPT e o comportamento esperado por um iniciante em programação:

```
if encontrada: #Codigo gerado pelo ChatGPT
if encontrada is True: #Codigo esperado de um iniciante
```

```
if x % i == 0: #Codigo gerado pelo ChatGPT

resultado = x % 1 #Codigo esperado de um iniciante
if (resultado == 0):
```

Operadores lógicos (AND, OR e NOT): O uso de operadores lógicos possui contornos que remetem às boas práticas de programação. No geral, a IA generativa evita a criação de condições complexas (com 2 ou mais operadores). Ao identificar tal necessidade, são gerados condições aninhadas, o que divide a lógica em partes menores. Por exemplo, no código abaixo era solicitado um algoritmo para um jogo de par ou ímpar. O ChatGPT criou uma estrutura de IF/ELSE, evitando assim o uso de múltiplos operadores lógicos em uma única condição:

```
soma = j1 + j2

if soma % 2 == 0:
```

```
    if p == 1:
        return "Jogador 1 ganha!"
    else:
        return "Jogador 2 ganha!"
```

Operadores relacionais ($<, >, <=$ e $>=$): O comportamento dos operadores relacionais também é similar ao uso dos operadores lógicos, isto é, evitam-se estruturas complexas, contendo múltiplos operadores. No exemplo abaixo, o enunciado do exercício solicitava a criação de um caça-palavras, sendo necessário identificar termos presentes nas diagonais. Enquanto isso pode ser desafiador por diversos fatores, o ChatGPT utilizou uma comparação de índices para resolver o problema.

```
if diagonal == palavra or diagonal[::-1] == palavra:
    if i == j:
        resultados.append(f"1 Palavra \"{palavra}\" na diagonal
           secundaria")
    elif i < j:
        resultados.append(f"2 Palavra \"{palavra}\" acima da
           diagonal secundaria")
    else:
        resultados.append(f"3 Palavra \"{palavra}\" abaixo da
           diagonal secundaria")
        encontrada = True
```

Laços de repetição (FOR/WHILE): O ChatGPT utiliza os laços de repetição como um importante instrumento para abstrair a lógica dos exercícios. Em linhas gerais, isso quer dizer que grande parte da lógica para solucionar um exercício encontra-se em um laço de repetição. Por exemplo, no enunciado de um exercício era exposto que o programa a ser desenvolvido deveria ler a entrada de dados do usuário e mostrar um processamento de acordo com cada entrada. O exercício, no entanto, não apresentava um limite de entradas e nem possíveis erros. Mesmo assim, o ChatGPT criou um *loop* infinito capaz de ser finalizado usando a instrução *break* e ainda fez o tratamento de dados usando o bloco *try/except*:

```
while True:
    try:
        condicao = input().strip()
        if condicao == "":
            break
        print(papagaio_linguagem(condicao))
    except EOFError:
        break
```

Em outro exercício, foi exposto que era necessário armazenar o ano e a força de diferentes alces. O ChatGPT criou uma *array* de objetos e usou um laço FOR para percorrê-lo, como ilustrado a seguir:

```
for ano, forca in alces:
```

Em ambos os casos, fica evidente a capacidade da IA generativa abstrair a lógica dos exercícios em laços de repetição que são utilizados juntos com outros recursos de programação (como o uso de objetos, *break* e *try/except*). Em união, isso representa que o ChatGPT é capaz unificar recursos de programação e laços de repetição para resolver problemas bem mais complexos. Esse comportamento é oposto ao de um(a) estudante que está aprendendo a programar, tendo em vista que muitos desses conceitos ainda não são compreendidos pelo aluno(a). Por exemplo, em um dos exercícios analisados era necessário mapear o tamanho de uma *string* e analisar cada caractere. Depois, deveria verificar a ocorrência desses caracteres em uma matriz. O ChatGPT encapsulou todo o processamento em uma única linha de código:

```
diagonal = ''.join(matriz[i + k][j - k].lower() for k in range(
    len(palavra)))
```

Ademais, o ChatGPT possui grande domínio sobre o uso de métodos e funcionalidades que podem facilitar o processamento do laço FOR. Por exemplo, o ChatGPT foi capaz de adotar o método *RANGE()* em múltiplos cenários, como exposto a seguir:

```
for _ in range(N): #RANGE com 1 parametro
for n in range(1, 21): #RANGE com 2 parametros
for _ in range(n + k - 2) #RANGE + operacoes matematicas
for i in range(len(alturas) - 1): #RANGE + LEN + operacao
    matematica
```

***Arrays*:** O ChatGPT possui muita facilidade para utilizar *arrays* e realizar operações de busca, identificação ou ordenamento. Mas, os trechos de códigos gerados por essa IA evidenciam práticas avançadas de programação. Um exemplo é o uso da função *map()* e do tipo de dados *tuple*, como exposto a seguir:

```
alces = [tuple(map(int, input().split())) for _ in range(n + k -
    2)]
```

Além do mais, como já foi exposto, é comum que o ChatGPT utilize *array* de objetos para resolver os problemas. Por exemplo, no código abaixo, o ChatGPT armazenou em cada posição de um *array* três dados distintos:

```
x1, y1, z1 = naves[i]
```

Funções: O ChatGPT também deriva o nome das funções conforme os enunciados dos exercícios. Por vezes, são utilizados identificadores significativos. No exemplo abaixo, o exercício solicitava criar um programa capaz de verificar palavras, o ChatGPT então gerou a seguinte função:

```
def verificar_palavra(palavra):
```

As funções geradas pelo ChatGPT são reduzidas, encapsuladas e bem organizadas. Diversas boas práticas de programação são observadas, como criação de funções que possuam apenas um objetivo e que são programadas como poucas linhas de código, evitando a complexidade, como exposto a seguir:

```
def quantidade_digitos(n, m):
    resultado = n ** m
    return len(str(resultado))
```

Em relação ao total de parâmetros utilizados, quase sempre são utilizados funções com possuem um número reduzido (3 ou menos parâmetros). No entanto, podem ser gerados códigos que possuem uma ampla lista de parâmetros, como exposto logo abaixo. Todavia, quando isso ocorre, o ChatGPT prioriza o objetivo da função, isto é, muitas vezes uma longa lista de parâmetros representa que o objetivo da função é realizar um cálculo matemático, o que necessita de diversas variáveis. Abaixo, é apresentado um exemplo disso:

```
def distancia(x1, y1, z1, x2, y2, z2):
    return math.sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2 + (z2 - z1)
        ** 2):
```

Por fim, o ChatGPT vai além do retorno primitivo de dados (como números inteiros ou status de verdadeiro/falso). Ao usar a instrução de retorno, a IA generativa pode realizar cálculos matemáticos (evitando assim a criação de novas variáveis), verificação lógica e retorno de objetos, como detalhado a seguir:

```
# retorno de objeto
return (coordenada_coluna, linha)

# retorno de calculo matematico
return 1 + N * (N - 1) * (N - 2) // 6

# retorno verificacao logica
return sum_of_divisors == x
```

Diante dos resultados apresentados, é possível avançar com a segunda QP deste estudo.

4.2. Quais elementos identificam que um exercício introdutório de programação foi presumidamente resolvido por uma IA generativa?

Os códigos gerados pelo ChatGPT, e provavelmente, por muitas IAs generativas, seguem padrões que contrastam com os códigos desenvolvidos por alunos que estão aprendendo a programar. [Medeiros et al. 2018], por exemplo, destacou que as habilidades de solução de problemas e a abstração são desafios comuns para iniciantes em programação. O ChatGPT, no entanto, foi capaz de contrastar isso, identificando pontos chaves no enunciado e aplicando-os para nomear funções e variáveis, além de conseguir planejar soluções que levavam em consideração a natureza do problema, dividindo-o em partes menores.

Outro conjunto de desafios comuns ao iniciantes em programação é a falta de habilidade matemática, aspecto já apontado por [Medeiros et al. 2018] e [Qian and Lehman 2017]. Como exposto, o ChatGPT é capaz de fazer operações matemáticas em diferentes níveis de complexidade, adotando isso até mesmo dentro das estruturas de condições e das instruções de retorno.

Conceito
Variável
• Nome das variáveis segue padronização? (exemplo: camelCase, snake_case) • Nome das variáveis são significativos? • A atribuição das variáveis é complexa? (exemplo: usando funções)
Estruturas de Decisão
• As estruturas de decisão estão isoladas em blocos com quebras de linha? • As estruturas de decisão estão indentadas corretamente? • Há estruturas aninhadas para evitar lógica complexa nas condições? • As condições possuem 2 ou menos operadores lógicos? • As condições possuem 2 ou menos operadores relacionais?
Estruturas de repetição
• Há uso de estruturas bidimensionais? • Há uso de métodos específicos da linguagem de programação? • Há criações complexas envolvendo operações e métodos? • Há uso da instrução *break* ou *continue*?
Arrays
• Há operações usando os índices? • Há uso de recursos avançados de programação, como *map* ou *set*? • Há uso de objetos dentro de *arrays*? • Há uso de *arrays* dentro de *arrays*?
Funções
• O nome da função segue padronização? • As funções possuem poucas linhas? • As funções possuem um único comportamento? • As funções possuem até 3 parâmetros? • As funções realizam operações de retorno complexas?
Geral
• O código está indentado corretamente? • Existe padronização entre os nomes das variáveis e das funções? • Existe quebra de linhas para isolar blocos de códigos? • O código está comentado?

Tabela 3. Síntese das principais características observadas em códigos gerados pelo ChatGPT

[Qian and Lehman 2017] também citam que muitos estudantes que estão aprendendo a programar possuem dificuldades em compreender aspectos elementares da programação, como o uso de variáveis e atribuição de valores e também a adoção de estruturas de condição e repetição. Isso não ocorreu nos códigos produzidos pelo ChatGPT. Ao contrário, a IA generativa teve grande domínio, adotando práticas avançadas de programação para esses recursos. Para auxiliar essa comparação, a Tabela 3 apresenta uma síntese dos elementos que identificam que o código gerado possivelmente foi produzido por uma IA generativa. Importante salientar que tais itens foram abstraídos de acordo com a linguagem Python. É provável que alguns padrões sejam repetidos e que novos padrões sejam identificados de acordo com linguagens distintas. Além do mais, essa análise, bem como os aspectos observados na primeira QP, são baseados nos padrões observados e na expectativa dos códigos que alunos que estão aprendendo a programar.

5. Discussão

Diante dos resultados apresentados, é essencial detalhar os achados para os professores de programação que irão avaliar resolução de exercícios introdutórios de programação. Em primeiro lugar, estudantes iniciantes em programação podem desenvolver bons códigos, o simples uso de um recurso avançado de programação (como uso de *tuple* ou *map*) não deve ser suficiente para classificar uma solução como sendo gerada por IA. No entanto, a ocorrência de diversas práticas avançadas de programação - sobretudo de conceitos ainda não explorados - podem demonstrar que o código não foi escrito por um aluno, mas sim por uma IA generativa. Nesse sentido, as características apresentadas na Tabela 3 deve ser adotada de acordo com cada contexto.

Em segundo lugar, os professores devem focar em aspectos referentes ao alto nível da programação. Elementos como a abstração, planejamento e pensamento crítico devem ser o foco dos professores. Para ensinar programação introdutória na era das IAs generativas muitas abordagens devem ser repensadas. Por exemplo, ao criar um enunciado no qual o estudante deve ordenar um vetor, o professor já deixa todo o processamento disponível para que uma IA resolva o problema. Ao invés disso, podem ser adotadas ideias inovadoras. Um exemplo é a proposta de [Denny et al. 2024a], na qual são apresentados desenhos (exemplo: Imagem 1 - desenho de um vetor desordenado e Imagem 2 - desenho de um vetor ordenado). Assim, o aluno deve verificar essa imagem, abstrair o problema e então codificar uma solução. Isso não resolve todos os problemas, mas é uma possível solução para mitigar o uso de IAs generativas.

Na prática, as IAs generativas oferecem uma janela de oportunidade para repensar o ensino e o aprendizado de programação introdutória. Nunca foi tão fácil resolver exercícios introdutórios de programação, mas, ao mesmo tempo, nunca foi tão difícil avaliar tais soluções.

6. Conclusão

Este artigo apresentou um estudo de caso sobre como uma IA generativa aborda e resolve problemas de programação introdutória. Para isso foi conduzido uma investigação na IA generativa mais usada por iniciantes em programação (ChatGPT), na qual 1.258 linhas de código foram inspecionadas pelos pesquisadores. Resultados demonstram diversas evidências do uso de IA generativa, que contrastam com o código de um aluno que está aprendendo a programar.

Limitações: Os resultados apresentados neste trabalho possuem escopo voltado ao ensino introdutório de programação utilizando Python, o que pode inviabilizar generalizações para outras linguagens (como C ou C++). Além do mais, o caso analisado foi o ChatGPT, embora existam diferentes IAs generativas capazes de escrever códigos. Para evitar a disseminação dessas ameaças, deve-se destacar que o Python e o ChatGPT foram selecionados por serem a escolha mais comum de iniciantes em programação, de acordo com o relatório mais recente produzido pelo [Stack Overflow 2023].

Trabalhos futuros: Replicar o estudo realizado em outras IAs generativas, comparando os códigos produzidos por alunos iniciantes em programação com os códigos produzidos por tais IAs e/ou solicitar que a IA generativa simplifique o código (evitando uso de funções, por exemplo). Outra linha de ação relevante é criar um software que utilize os dados da Tabela 3 para identificar códigos e/ou trechos que presumidamente foram produzidos por uma IA generativa.

Agradecimentos

O presente trabalho foi realizado com apoio da Coordenação de Aperfeiçoamento de Pessoal de Nível Superior – Brasil (CAPES) – Código de Financiamento 001 e processo nº 2019/26871-4, Fundação de Amparo à Pesquisa do Estado de São Paulo (FAPESP).

Referências

[Biderman and Raff 2022] Biderman, S. and Raff, E. (2022). Fooling moss detection with pretrained language models. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 2933–2943, New York, NY, USA. Association for Computing Machinery.

[Biswas 2023] Biswas, S. (2023). Role of chatgpt in computer programming.: Chatgpt in computer programming. *Mesopotamian Journal of Computer Science*, 2023:8–16.

[Bull and Kharrufa 2023] Bull, C. and Kharrufa, A. (2023). Generative ai assistants in software development education: A vision for integrating generative ai into educational practice, not instinctively defending against it. *IEEE Software*.

[Denny et al. 2024a] Denny, P., Leinonen, J., Prather, J., Luxton-Reilly, A., Amarouche, T., Becker, B. A., and Reeves, B. N. (2024a). Prompt problems: A new programming exercise for the generative ai era. In *Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1*, pages 296–302.

[Denny et al. 2024b] Denny, P., Prather, J., Becker, B. A., Finnie-Ansley, J., Hellas, A., Leinonen, J., Luxton-Reilly, A., Reeves, B. N., Santos, E. A., and Sarsa, S. (2024b). Computing education in the era of generative ai. *Commun. ACM*, 67(2):56–67.

[Euchner 2023] Euchner, J. (2023). Generative ai. *Research-Technology Management*, 66(3):71–74.

[Feuerriegel et al. 2024] Feuerriegel, S., Hartmann, J., Janiesch, C., and Zschech, P. (2024). Generative ai. *Business & Information Systems Engineering*, 66(1):111–126.

[Filho et al. 2023] Filho, L. P., Souza, T., and Paula, L. (2023). Análise das respostas do chatgpt em relação ao conteúdo de programação para iniciantes. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1738–1748, Porto Alegre, RS, Brasil. SBC.

[Imran and Almusharraf 2023] Imran, M. and Almusharraf, N. (2023). Analyzing the role of chatgpt as a writing assistant at higher education level: A systematic review of the literature. *Contemporary Educational Technology*, 15(4):ep464.

[Liu et al. 2024] Liu, Y., Le-Cong, T., Widyasari, R., Tantithamthavorn, C., Li, L., Le, X.-B. D., and Lo, D. (2024). Refining chatgpt-generated code: Characterizing and mitigating code quality issues. *ACM Trans. Softw. Eng. Methodol.* Just Accepted.

[Medeiros et al. 2018] Medeiros, R. P., Ramalho, G. L., and Falcão, T. P. (2018). A systematic literature review on teaching and learning introductory programming in higher education. *IEEE Transactions on Education*, 62(2):77–90.

[Oliveira et al. 2023] Oliveira, K. K. d. S., Marcolino, A. d. S., Deus, W. S. d., Falcão, T. P. d. R., and Barbosa, E. F. (2023). Pensamento computacional na programação introdutória e habilidades do século xxi: um mapeamento sistemático da literatura. *Revista Novas Tecnologias na Educação-RENOTE*, 21(2):519–531.

[Qian and Lehman 2017] Qian, Y. and Lehman, J. (2017). Students' misconceptions and other difficulties in introductory programming: A literature review. *ACM Transactions on Computing Education (TOCE)*, 18(1):1–24.

[Sharples 2023] Sharples, M. (2023). Towards social generative ai for education: theory, practices and ethics. *Learning: Research and Practice*, 9(2):159–167.

[Stack Overflow 2023] Stack Overflow, D. S. (2023). 2023 developer survey: Ai search tools.

[Taecharungroj 2023] Taecharungroj, V. (2023). "what can chatgpt do?" analyzing early reactions to the innovative ai chatbot on twitter. *Big Data and Cognitive Computing*, 7(1):35.

[Tew and Guzdial 2010] Tew, A. E. and Guzdial, M. (2010). Developing a validated assessment of fundamental cs1 concepts. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education*, page 97–101, Milwaukee, Wisconsin, USA. Proceedings [...]. New York, NY, USA. Association for Computing Machinery.

[Virvou et al. 2023] Virvou, M., Tsihrintzis, G. A., Sotiropouloss, D. N., Chrysafiadi, K., Sakkopoulos, E., and Tsichrintzi, E.-A. (2023). Chatgpt in artificial intelligence-empowered e-learning for cultural heritage: The case of lyrics and poems. In *2023 14th International Conference on Information, Intelligence, Systems & Applications (IISA)*, pages 1–9. IEEE.

[Yin 2015] Yin, R. K. (2015). *Estudo de Caso-: Planejamento e métodos*. Bookman editora.