INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO SERTÃO PERNAMBUCANO

COORDENAÇÃO DO CURSO DE SISTEMAS PARA INTERNET

SISTEMAS PARA INTERNET

FELIPE BEZERRA DE SOUZA FREIRE

O AVANÇO DO GPT-4 EM RELAÇÃO AO GPT-3 NO DESENVOLVIMENTO DE SOFTWARE

SALGUEIRO

2025

Dados Internacionais de Catalogação na Publicação (CIP)

F866 Freire, Felipe Bezerra de Souza.
O Avanço do GPT-4 em Relação ao GPT-3 no Desenvolvimento de Software / Felipe Bezerra de Souza Freire. - Salgueiro, 2025.
32 f.

Trabalho de Conclusão de Curso (Sistemas para Internet) -Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, Campus Salgueiro, 2025.
Orientação: Profª. Msc. Francenila Rodrigues Junior.

1. Inteligência artificial. 2. Desenvolvimento de Software. 3. Processamento de Linguagem Natural. 4. GPT-3. 5. GPT-4. I. Título.

CDD 006.3

Gerado automaticamente pelo sistema Geficat, mediante dados fornecidos pelo(a) autor(a)

2025

FELIPE BEZERRA DE SOUZA FREIRE

O AVANÇO DO GPT-4 EM RELAÇÃO AO GPT-3 NO DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado a Coordenação do curso de Sistemas para Internet do Instituto Federal de Educação, Ciência e Tecnologia do Sertão Pernambucano, campus Salgueiro, como requisito parcial à obtenção do título de Tecnólogo em Sistemas para Internet.

Aprovado em: 26/09/2025.

BANCA EXAMINADORA

_____

Prof. Francenila Rodrigues Junior **Orientador(a)**
IF Sertão PE – Campus Salgueiro

_____

Prof. Francisco Junio da Silva Fernandes
IF Sertão PE – Campus Salgueiro

_____

Prof. Gustavo Freitas Sanchez
IF Sertão PE – Campus Salgueiro

SALGUEIRO 2025

AGRADECIMENTOS

A Deus, pela presença constante em minha vida, concedendo-me força, sabedoria e serenidade para superar os desafios desta jornada acadêmica. Sem Sua luz e proteção, não seria possível chegar até aqui e concluir mais esta importante etapa da minha formação.

Aos meus pais, José Roberto e Cleidivania Bezerra, por todo o amor, dedicação e exemplo de honestidade e trabalho. Cada conquista é resultado dos valores que me transmitiram e do apoio incondicional que sempre encontrei em vocês.

Ao meu orientador, professora Francenila Rodrigues, pela orientação técnica, paciência e dedicação durante todo o desenvolvimento desta pesquisa. Sua experiência e comprometimento foram essenciais para o amadurecimento e a conclusão deste trabalho.

A professora Patrícia, pelo apoio e pelas valiosas contribuições nas etapas iniciais do projeto, que foram fundamentais para a sua estruturação e aperfeiçoamento.

Por fim, agradeço a todos os colegas, amigos e professores que contribuíram para a realização deste trabalho, seja com palavras de incentivo, compartilhamento de conhecimento ou simples gestos de apoio ao longo desta caminhada.

O AVANÇO DO GPT-4 EM RELAÇÃO AO GPT-3 NO DESENVOLVIMENTO DE SOFTWARE

Felipe Bezerra de S. Freire[1], Francenila Rodrigues Junior S.[2]

Instituto Federal de Educação Ciência e Tecnologia do Sertão Pernambucano BR 232, Km 504, sentido Recife, Zona Rural, 56000-000 – Salgueiro/ PE – Brasil

felipe.freire@aluno.ifsertao-pe.edu.br[1],

francenila.rodrigues@ifsertao-pe.edu.br[2]

Abstract. *With the fast-paced advancement of Artificial Intelligence, tools powered by language models such as GPT-3 and, more recently, GPT-4, have increasingly become part of programmers' daily routines during software development. This study presents a comparative analysis between these two models to understand when compared to its predecessor. The research combines a theoretical review with practical tests using code examples in various languages, aiming to observe how the models respond to prompts, explain their solutions, and assist in debugging. The results show that GPT-4 stands out for offering more accurate outputs, a better understanding of context, and clearer communication, which can positively influence both productivity and learning.*

Resumo. *Com o crescimento acelerado da Inteligência Artificial, ferramentas baseadas em modelos de linguagem como o GPT-3 e, mais recentemente, o GPT-4, começaram a fazer parte da rotina de programadores durante o desenvolvimento de software. O presente trabalho propõe uma análise comparativa entre esses dois modelos, com o intuito de compreender como o GPT-4 tem ampliado o apoio oferecido ao programador em relação ao seu antecessor. A pesquisa se baseia em revisão bibliográfica e em testes práticos realizados com exemplos de código em diferentes linguagens, buscando observar como os modelos respondem a instruções, explicam soluções e ajudam na correção de erros. Os dados analisados indicam que o GPT-4 se destaca por entregar respostas mais precisas, maior compreensão do contexto e uma comunicação mais clara, o que pode impactar positivamente na produtividade e no aprendizado de quem programa.*

LISTA DE ABREVIAÇÕES E SIGLAS

API	Interface de Programação das Aplicações
AWS	Amazon Web Services
GPT	Generative Pre-trained Transformer
IA	Inteligência Artificial
IDE	Integrated Development Environment
PLN	Processamento de Linguagem Natural
REPL	Read-Eval-Print Loop
VSCode	Visual Studio Code

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1. Introdução

O rápido avanço da Inteligência Artificial (IA) tem causado mudanças significativas na área da programação. À medida que ferramentas cada vez mais sofisticadas ganham espaço no cotidiano dos desenvolvedores, uma dúvida recorrente começa a surgir: a IA veio para auxiliar ou substituir o trabalho humano na codificação? Segundo Ferreti, Salesi e Cavichiolli (2023), esse debate envolve não apenas aspectos técnicos, mas também questões como autonomia criativa e o impacto da automação sobre a produtividade. Nesse contexto, modelos de linguagem como o GPT-3 e o mais recente GPT-4 tornaram-se centrais para essa discussão, representando marcos importantes na forma como interagimos com o código.

Lançado em 2020, o GPT-3 chamou atenção por sua capacidade de processar informações com base em 175 bilhões de parâmetros, o que marcou um avanço expressivo na geração automática de texto e código (Brown et al., 2020). Três anos depois, a OpenAI apresenta o GPT-4, que, segundo a própria organização, incorporou melhorias significativas em compreensão de contexto e passou a operar com entradas multimodais como imagens e texto simultaneamente (OpenAI, 2023). Esses avanços têm transformado a forma como programadores interagem com o código, oferecendo suporte técnico cada vez mais sofisticado. No entanto, como aponta essa evolução, surgem também dúvidas sobre qual será, de fato, o papel do programador diante de um ambiente cada vez mais automatizado.

De acordo com Brown et al. (2020), o GPT-3 apresentou resultados expressivos na geração automática de texto e código, o que passou a oferecer aos programadores uma alternativa ágil para lidar com tarefas rotineiras. Com o lançamento do GPT-4, essas capacidades foram significativamente ampliadas. Segundo a OpenAI (2023), o novo modelo é capaz de interpretar instruções mais complexas, entender melhor o contexto e fornecer respostas mais claras e precisas. Como reflexo dessa evolução, ferramentas práticas como o GitHub Copilot, que utilizam esses modelos, têm sido cada vez mais incorporadas ao cotidiano do desenvolvimento de software (GitHub, 2022).

Apesar dos avanços trazidos por essas tecnologias, é preciso reconhecer que elas também apresentam desafios importantes. Bender et al. (2021) alertam para riscos relacionados à confiabilidade das respostas geradas por modelos de linguagem, incluindo a possibilidade de propagação de erros e o surgimento de uma dependência excessiva por parte dos programadores. Além disso, como destacam Floridi et al. (2018), questões éticas envolvendo autoria de código, originalidade e o impacto da IA no futuro da profissão têm ganhado espaço nas discussões acadêmicas e profissionais.

Considerando esse cenário de avanços e desafios, o presente trabalho tem como proposta realizar uma análise comparativa entre os modelos de linguagem GPT-3 e GPT-4, com foco no suporte que oferecem ao programador durante o desenvolvimento de software. Busca-se compreender de que forma o GPT-4 aprimorou aspectos como a qualidade das respostas, a compreensão de contexto e a capacidade de auxiliar na correção de erros, em relação ao seu antecessor.

Para alcançar esse objetivo, serão conduzidas uma revisão bibliográfica e uma série de testes práticos com exemplos de código. A partir dessas análises, pretende-se identificar os benefícios, limitações e impactos reais do uso de cada modelo no cotidiano do programador.

Objetivo Geral: Avaliar o avanço do modelo GPT-4 em comparação ao GPT-3 no apoio ao desenvolvimento de software.

Objetivos Específicos:

- Investigar a qualidade das respostas fornecidas por ambos os modelos;
- Analisar a compreensão contextual em tarefas de programação;
- Avaliar a eficácia dos modelos na correção de erros de código.

Metodologia: O estudo adota uma abordagem mista, combinando revisão bibliográfica sobre os modelos GPT e testes práticos com geração de códigos. A avaliação será feita com base na clareza, precisão e aplicabilidade das respostas em cenários reais de desenvolvimento.

O tema é relevante porque ferramentas de IA como GPT-3 e GPT-4 já fazem parte do cotidiano do desenvolvimento de software. Comparar as duas versões ajuda a entender ganhos e limites no apoio à programação, como clareza das respostas, depuração e produtividade. No contexto da formação acadêmica em tecnologia que inclui disciplina de Inteligência Artificial com noções de funcionamento e treinamento de modelos, os resultados oferecem referência prática para o uso responsável dessas ferramentas em atividades e projetos, aproximando a aprendizagem das demandas do mercado de trabalho.

Além desta introdução, o trabalho está dividido em sete seções principais. A Seção 2 apresenta os conceitos fundamentais, abordando Inteligência Artificial, Processamento de Linguagem Natural e os modelos GPT. A Seção 3 detalha a metodologia utilizada. A Seção 4 traz os resultados obtidos nos testes. A Seção 5 apresenta a análise comparativa e a discussão dos achados. A Seção 6 reúne as conclusões e considerações finais. Por fim, a Seção 7 expõe as propostas de trabalhos futuros.

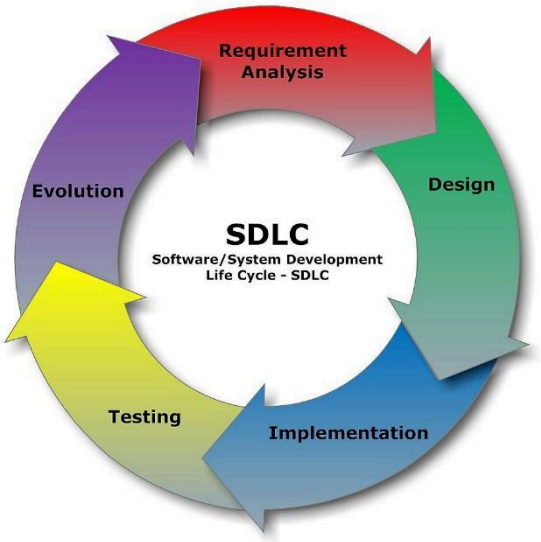
2. Fundamentação Teórica

2.1. Inteligência Artificial no Desenvolvimento de Software

A Inteligência Artificial (IA) tem assumido um papel cada vez mais relevante no desenvolvimento de software, promovendo inovações que impactam diretamente a produtividade, a precisão e a eficiência das soluções criadas. Segundo diversos autores, a IA vem sendo amplamente utilizada para automatizar tarefas repetitivas, como a geração de código, correção de erros e elaboração de testes automatizados, liberando os programadores para se dedicarem a atividades mais estratégicas e criativas no processo de desenvolvimento.

Além de acelerar a produção, a IA também contribui para a tomada de decisões, ao oferecer análises e sugestões com base em grandes volumes de dados. Tecnologias como assistentes virtuais, chatbots e sistemas de recomendação são exemplos concretos de aplicações impulsionadas por IA, que enriquecem a experiência do usuário e abrem espaço para novas funcionalidades nos sistemas. Esse processo pode ser visualizado no ciclo de vida do desenvolvimento de software, ilustrado na figura 1.

Figura 1 - Ciclo de vida do desenvolvimento de software



Fonte: WIKIMEDIA COMMONS (2023).

No entanto, a adoção da IA em ambientes de desenvolvimento não está isenta de desafios. Entre os principais pontos de atenção, destacam-se os riscos de viés nos dados de treinamento, a dificuldade de interpretar ou justificar determinadas decisões tomadas pelo sistema e as vulnerabilidades de segurança que podem ser exploradas em modelos automatizados.

2.2. Processamento de Linguagem Natural (PLN)

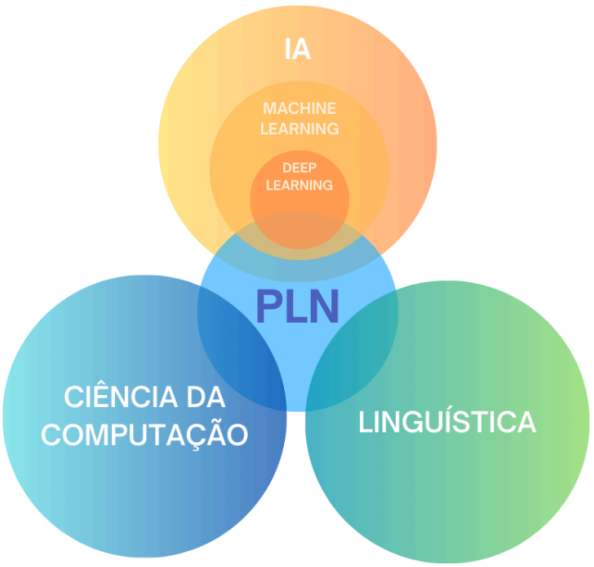
O Processamento de Linguagem Natural (PLN) é uma área da inteligência artificial que busca viabilizar a comunicação entre seres humanos e computadores por meio da linguagem escrita ou falada. De acordo com especialistas da área, o principal objetivo do PLN é fazer com que os sistemas computacionais sejam capazes de compreender, interpretar e gerar linguagem humana de maneira eficaz e funcional.

As aplicações práticas do PLN são amplas e estão cada vez mais presentes no cotidiano. Tecnologias como tradução automática, análise de sentimentos, assistentes virtuais, chatbots e reconhecimento de fala são exemplos claros de como o PLN transforma a interação entre pessoas e máquinas, tornando-a mais natural e acessível.

Historicamente, os primeiros estudos em PLN surgiram na década de 1950, quando pesquisadores começaram a explorar métodos automáticos de tradução de textos. Desde então, a área evoluiu de forma expressiva, incorporando técnicas avançadas de aprendizado de máquina e redes neurais, o que tem aumentado significativamente a precisão e a eficiência dos sistemas.

Como mostra a Figura 2, o Processamento de Linguagem Natural (PLN) integra o campo da Inteligência Artificial (IA), tendo o aprendizado de máquina (ML) como técnica predominante para os modelos recentes (JURAFSKY; MARTIN, 2025).

Figura 2 – Interseção entre áreas relacionadas ao Processamento de Linguagem Natural



Fonte: JURAFSKY; MARTIN (2025).

O Processamento de Linguagem Natural (PLN) é uma subárea da IA que se concentra na interação entre computadores e a linguagem humana. O objetivo do PLN é permitir que as máquinas compreendam, interpretem e gerem linguagem de maneira natural e significativa.

As aplicações de PLN são vastas e incluem:

- **Análise de sentimentos:** Avaliação de opiniões em redes sociais e avaliações de produtos.
- **Tradução automática:** Conversão de texto de um idioma para outro.

Assistentes virtuais: Sistemas como Siri, Alexa e Google Assistant que interagem com usuários.

- **Chatbots:** Programas que simulam conversas humanas para atendimento ao cliente.

2.3. GPT-3: Características e Limitações

Lançado pela OpenAI em 2020, o modelo GPT-3 (Generative Pre-trained Transformer 3) representou um marco significativo na área da inteligência artificial generativa. Segundo Brown et al. (2020), com seus 175 bilhões de parâmetros treinados a partir de uma imensa base de dados da internet, o GPT-3 demonstrou a capacidade de gerar textos coerentes, traduzir idiomas, resumir conteúdos, responder perguntas e até mesmo escrever código com base em instruções simples em linguagem natural. Essa versatilidade contribuiu para a popularização do uso de IA em atividades técnicas, como o desenvolvimento de software.

Uma de suas características mais marcantes está no desempenho em tarefas de *few-shot learning*, em que o modelo consegue generalizar a partir de poucos exemplos fornecidos no próprio prompt. Ainda segundo Brown et al. (2020), essa habilidade o diferencia de modelos anteriores, que exigiam grandes volumes de dados rotulados para alcançar bons resultados. Outro avanço importante foi a adoção do aprendizado por reforço com feedback humano (RLHF), o que contribuiu para respostas mais úteis em interações do tipo conversacional, como no ChatGPT.

Apesar de suas capacidades impressionantes, o GPT-3 possui limitações relevantes. Bender et al. (2021) alertam que o modelo não compreende o conteúdo de forma real, mas apenas realiza previsões com base em padrões linguísticos. Isso pode resultar em respostas factualmente incorretas, inconsistentes ou até tendenciosas, especialmente quando o prompt remete a dados sensíveis ou ambíguos. Esse comportamento gera preocupação sobre a confiabilidade de suas respostas, sobretudo em contextos que exigem lógica apurada ou conhecimento técnico específico.

No contexto do desenvolvimento de software, Souza (2023) avaliou a performance do GPT-3 em 100 problemas de programação das plataformas LeetCode e BeeCrowd. Os resultados indicaram que o modelo foi capaz de resolver corretamente 71% dos desafios, mas teve dificuldades notáveis em problemas mais complexos, principalmente aqueles que envolviam múltiplas etapas ou exigiam otimização de código. Segundo o autor, isso evidencia a necessidade de validação humana cuidadosa no uso do modelo em tarefas práticas.

Outra limitação importante está na ausência de capacidades multimodais. O GPT-3 não é capaz de processar imagens ou qualquer entrada que não seja textual, o que compromete seu desempenho em problemas visuais, como os encontrados em algumas atividades do BeeCrowd. Essa deficiência só foi superada na versão seguinte, o GPT-4, que passou a aceitar entradas combinadas de texto e imagem (OpenAI, 2023).

2.4. GPT-4: Avanços, Multimodalidade e Melhorias Contextuais

A evolução do GPT-3 para o GPT-4 representou mais do que um aprimoramento técnico, foi uma mudança significativa na aplicação prática dos modelos de linguagem, especialmente no desenvolvimento de software. De acordo com a OpenAI (2023), uma das principais melhorias do GPT-4 está em sua capacidade de interpretar comandos com profundidade contextual, tornando-o mais eficaz em tarefas que exigem raciocínio estruturado, lógica sequencial e compreensão de múltiplas instruções.

Embora essa funcionalidade ainda dependa de plataformas compatíveis, sua presença já indica um novo patamar na interação entre programador e IA. Além disso, o GPT-4 apresenta uma sensibilidade contextual superior: compreende melhor perguntas compostas, interpreta nuances de linguagem ambígua e mantém coerência em interações prolongadas, o que reduz a necessidade de reformular instruções, uma limitação frequente no GPT-3.

Em estudo realizado por Zhang et al. (2023), que analisou a performance dos dois modelos em problemas de lógica e algoritmos, o GPT-4 demonstrou um desempenho 17% superior ao GPT-3 em tarefas com múltiplas etapas e estruturas condicionais complexas. Os resultados reforçam a ideia de que o novo modelo oferece maior robustez técnica e confiabilidade.

Outro diferencial importante é a capacidade do GPT-4 de lidar com tarefas compostas dentro de um único prompt. Em situações em que o usuário solicita várias ações, como calcular a média de notas, salvar os dados em JSON e enviar para uma API REST, o GPT-4 tende a atender a todas as instruções de forma integrada e coesa. O GPT-3, por outro lado, frequentemente ignora partes da tarefa ou realiza a execução de forma incompleta.

3. Metodologia

3.1. Tipo de Pesquisa

Este estudo adota uma abordagem qualitativa e comparativa, com o objetivo de analisar o desempenho de dois modelos de linguagem natural baseados em inteligência artificial: GPT-3 e GPT-4. A análise é realizada por meio de experimentos envolvendo a geração de código em linguagem Python, aplicados em um cenário prático dentro do jogo Minecraft. O foco da comparação está em aspectos como clareza, precisão, estrutura e funcionalidade das respostas fornecidas por cada modelo..

O Minecraft foi escolhido como ambiente experimental por permitir execução em modo single-player offline, garantindo controle total do cenário e repetibilidade dos testes em uma máquina local, sem depender de conexões externas ou afetar outros jogadores. Além disso, possibilita configurar recursos de forma estável (inventário, posição inicial e terreno plano), o que assegurou consistência nas execuções. A escolha também se justifica pela acessibilidade e pela popularidade do jogo em contextos educacionais, o que o torna um ambiente didático apropriado para experimentos em programação, unindo praticidade técnica e relevância pedagógica.

Com o objetivo de verificar a estabilidade dos resultados, cada teste foi executado mais uma vez com os mesmos parâmetros e condições dentro do jogo. A repetição reproduziu os mesmos comportamentos e resultados do primeiro teste, reforçando a consistência dos resultados, já que não houve alteração no jogo nem no código.

3.2. Ferramentas e tecnologias utilizadas

Para a realização dos testes, foram utilizadas ferramentas amplamente conhecidas no desenvolvimento de software, incluindo ChatGPT (GPT-3 e GPT-4), Minecraft Java Edition, Python 3.11 com a biblioteca *pyautogui*, além do editor Visual Studio Code em ambiente Windows 10, conforme mostra a sequência na figura 3.

Figura 3 - Ferramentas e tecnologias utilizadas no ambiente de teste

Fonte: Elaborado pelo autor (2025)

A Figura 4 apresenta a padronização adotada no ambiente de testes, destacando a configuração inicial do personagem no jogo e os itens do inventário utilizados. O personagem do Minecraft foi posicionado manualmente no ponto inicial da estrada. O bloco de pedra foi alocado no campo do inventário 1 e a tocha, no campo 2, como destacado na Figura 4.

Figura 4 - Ferramentas e tecnologias utilizadas no ambiente de teste

Fonte: Captura de tela do autor (2025)

3.3. Descrição do Experimento

A atividade prática proposta aos modelos consistiu na geração de um código em Python, utilizando a biblioteca *pyautogui*, com o objetivo de automatizar a construção de uma estrada simples no jogo Minecraft. A estrada deveria ter 20 blocos de comprimento e atender aos seguintes critérios:

- O material da estrada deveria ser de pedra;
- Uma tocha deveria ser colocada em ambos os lados da estrada a cada 5 blocos;
- O código deveria ser escrito em Python 3;

Para garantir igualdade nas condições de teste, o mesmo prompt foi enviado aos modelos GPT-3 e GPT-4, conforme é mostrado a seguir: “Escreva um código em Python 3 usando a biblioteca *pyautogui* para automatizar a construção de uma estrada no Minecraft Java Edition. O mundo está em modo criativo, plano. O bloco de pedra está no campo do inventário 1, a tocha está no campo 2. O jogador já está posicionado no início da estrada. A estrada deve ter 20 blocos de comprimento. A cada 5 blocos, deve-se colocar uma tocha de cada lado. O código deve simular as ações do jogador com precisão, alternando entre os campos 1 e 2 do inventário e utilizando o mouse e teclado conforme necessário.”

Antes da execução do código, o ambiente foi configurado sequencialmente da seguinte maneira:

1. O mundo foi gerado no modo Criativo, terreno totalmente plano, com o jogador posicionado manualmente no ponto inicial da estrada;
2. O bloco de pedra foi posicionado no campo 1 do inventário;
   As tochas foram colocadas no campo 2 do inventário;
3. O código gerado foi salvo com extensão *.py* e executado no Minecraft em tela cheia. O script controlava o mouse e o teclado com *pyautogui*, simulando os comandos manuais de movimentação e construção no jogo.

Esse procedimento foi padronizado para garantir consistência entre os testes.

Para cada modelo, foi observado o seguinte:

- Se o código fornecido rodava tranquilamente sem erros;
- Se os blocos e tochas foram posicionados corretamente e como esperado;
- Se o comportamento visual era coerente com a tarefa solicitada;
- O tempo aproximado de resposta de cada modelo;
- A clareza e organização do código (variáveis, funções e organização a partir de comentários).

As execuções foram documentadas com capturas de tela em três momentos distintos do experimento: o envio do prompt e a resposta gerada por cada modelo de IA (GPT-3 e GPT-4), a visualização do código colado e salvo no editor VSCode, e, por fim, a execução prática no ambiente do Minecraft, com a estrada finalizada conforme as instruções do teste.

3.4. Critérios de Avaliação

A comparação entre os modelos foi realizada com base nos seguintes critérios descritos na Tabela 1 a seguir:

Tabela 1 - Critérios de avaliação

Critérios	Descrição
Clareza do código	Avalia se o código é de fácil compreensão, com nomes de variáveis descritivos e boa legibilidade.
Precisão da Execução	Verifica se o código realiza exatamente as ações solicitadas no enunciado da tarefa.
Necessidade de correções	Indica se foram necessárias edições ou ajustes manuais para que o código funcionasse corretamente.
Tempo de geração	Mede o tempo médio que cada modelo levou para gerar a resposta ao prompt enviado.
Organização e estrutura	Analisa o uso de boas práticas de programação, como modularização em funções, comentários e indentação adequada

Fonte: Autor

4. Resultados e Discussões

Este capítulo apresenta os resultados obtidos nos testes práticos realizados com os modelos GPT-3 e GPT-4, seguidos de uma análise comparativa entre ambos e da experiência do programador durante o processo. A proposta prática consistiu na geração de um código em Python, utilizando a biblioteca *pyautogui*, para simular a automação da construção dentro do jogo Minecraft, com a construção de uma estrada de 20 blocos e a colocação de tochas laterais a cada 5 blocos. Os resultados foram documentados com capturas de tela e avaliados com base em critérios técnicos definidos na metodologia.

4.1. Análise do desempenho do GPT-3

Esta seção apresenta a avaliação prática do desempenho do modelo GPT-3 na geração de um código em Python para automatizar a construção de uma estrada no jogo Minecraft, conforme descrito anteriormente na metodologia.

O quadro 1 mostra que a resposta do modelo consistiu em um código funcional básico, que utilizou um laço "for" para iterar ao longo do comprimento da estrada, posicionando blocos de pedra. A lógica para inserção das tochas também foi incluída, embora de forma incompleta: apesar de tentar posicionar uma tocha em cada lado a cada cinco blocos, não obteve sucesso na ação, apenas realizou os movimentos para esquerda e direita.

Quadro 1 - Destaque do código gerado pelo GPT-3 para o experimento no Minecraft

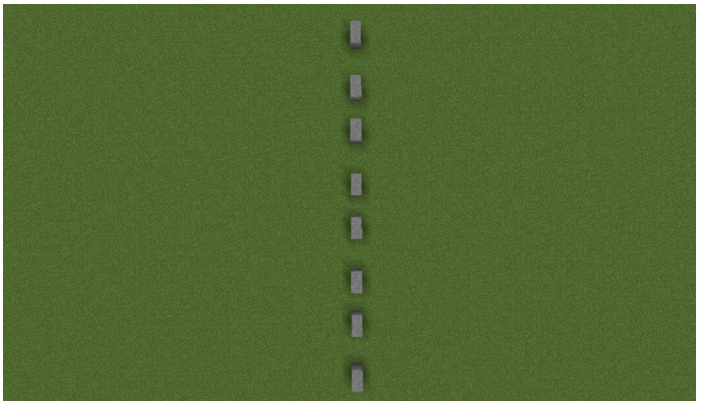
```
for i in range(1, 21):
    select_slot(1)
    place_block()
    if i % 5 == 0:
        turn_player(-90)
        select_slot(2)
        place_block()
        turn_player(180)
        place_block()
        turn_player(-90)
        select_slot(1)
    press_w()
```

Fonte: Elaborado pelo autor com base em resposta do GPT-3 (2025)

O código foi inserido no editor VSCode, salvo e executado sem ajustes manuais, iniciando-se a automação. No entanto, durante a execução, uma série de falhas visuais: o personagem realizava giros de 360°, travava em determinados pontos e os blocos eram posicionados de forma irregular frequentemente com saltos de dois espaços entre um bloco e

outro. Nenhuma tocha foi colocada, devido a erros de movimentação, visto abaixo na Figura 5.

Figura 5 - Execução do código gerado pelo GPT 3 no Minecraft



Fonte: Captura de tela no minecraft elaborado pelo autor

Apesar de ter conseguido gerar uma estrutura básica, o GPT-3 não atendeu integralmente aos critérios definidos no experimento. A estrada foi parcialmente construída, e o modelo demonstrou limitações claras na interpretação de instruções compostas, bem como na execução precisa das ações propostas.

O GPT-3 demonstrou capacidade de gerar um código funcional, mas com limitações claras na compreensão das instruções. A omissão de parte da tarefa (colocar tochas nos dois lados) revela uma interpretação superficial. O código é direto, mas pouco estruturado, sendo necessário realizar alterações manuais ou até mesmo que sejam enviados novos prompts para chegar no resultado esperado para o desafio.

Diante das falhas, o modelo oferece uma base pouco viável para desenvolvimento, especialmente em tarefas de automações e com apoio humano para validação e ajustes finais.

4.2. Análise do desempenho do GPT-4

Esta seção apresenta a avaliação prática do modelo GPT-4 ao executar a mesma tarefa atribuída ao GPT-3. Para essa análise, buscou-se observar não apenas a capacidade do modelo em reproduzir a lógica solicitada, mas também a clareza estrutural e a organização do código gerado. Como parte dessa etapa, é apresentado a seguir um trecho representativo da solução produzida pelo GPT-4, destacado no quadro 2, que ilustra a forma como o modelo estruturou funções, comentários e comandos de execução no contexto do experimento no Minecraft

Quadro 2 - Destaque do código gerado pelo GPT-4 para o experimento no Minecraft

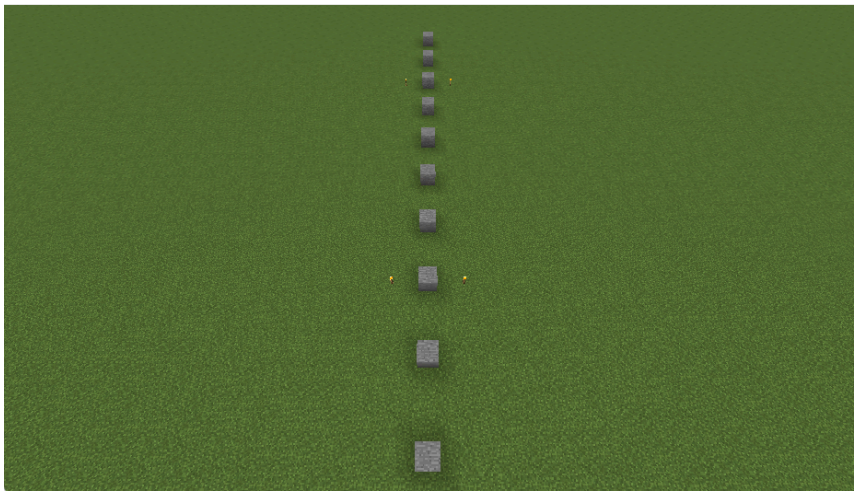
```
def construir_estrada():
    time.sleep(5)                # Aguarda 5s para o jogador se preparar
    selecionar_slot(1)           # Seleciona o bloco de pedra (slot 1)

    for i in range(20):          # Constrói 20 blocos de estrada
        colocar_bloco()          # Coloca um bloco à frente
        if (i + 1) % 5 == 0:     # A cada 5 blocos...
            virar_e_colocar_tocha(-1)  # Tocha à esquerda
            virar_e_colocar_tocha(1)   # Tocha à direita
        mover_para_frente()      # Anda um bloco

def virar_e_colocar_tocha(direcao):
    pyautogui.moveRel(direcao * 100, 0)
    selecionar_slot(2)           # Seleciona tocha
    colocar_bloco()
    selecionar_slot(1)           # Volta para a pedra
    pyautogui.moveRel(-direcao * 100, 0)
```

Fonte: Elaborado pelo autor com base em resposta do GPT-4 (2025)

A resposta fornecida pelo GPT-4 foi mais estruturada e tecnicamente refinada. O modelo organizou o código por meio de funções, utilizou nomes de variáveis descritivos e incluiu comentários explicativos que facilitam a leitura e compreensão do script. A lógica implementada contemplou parcialmente correto tanto o posicionamento dos blocos de pedra quanto das tochas, com rotações laterais em ambos os sentidos da estrada.

O script foi inserido no editor VSCode e executado sem necessidade de ajustes manuais, o que reforça a qualidade e precisão da solução proposta. Durante a execução, constatou-se uma movimentação fluida e coordenada: os blocos de pedra foram posicionados em sequência estável e, a cada 3 e 5 blocos, foram inseridas tochas corretamente dos dois lados, porém a estrada não conteve os 20 blocos, apenas 10, como destacado na Figura 6.

Figura 6 - Execução do código gerado pelo GPT 4 no Minecraft



Fonte: Captura de tela no minecraft elaborado pelo autor

A execução ocorreu sem travamentos, com rotações suaves e posicionamento adequado de alguns materiais utilizados. Embora nem todas as tochas tenham sido colocadas ao longo da extensão da estrada, o desempenho geral foi satisfatório e alinhado aos critérios definidos no experimento, mesmo que não tenha executado 100% correto ao que se foi pedido no desafio.

O GPT-4 demonstrou desempenho superior ao GPT-3 na realização da tarefa proposta. O código gerado foi mais completo, organizado e funcional, com interpretação correta da maior parte dos elementos do enunciado. A quantidade pequena de erros e a clareza estrutural tornam o código mais confiável e reutilizável. Este resultado reforça a evolução do modelo em termos de compreensão contextual por conta dos comentários, detalhamento técnico e adequação prática para uso por programadores.

4.3. Comparativo direto entre os modelos

Esta seção apresenta uma comparação objetiva entre os modelos GPT-3 e GPT-4, com base nos testes práticos descritos na metodologia e avaliados, segundo critérios técnicos que foram previamente definidos. Os aspectos analisados incluem clareza do código, precisão na execução, necessidade de correções (As correções deveriam aparecer somente caso o código não fosse capaz de executar o desafio no minecraft perante alguma alteração manual, caso execute sem nenhuma falha técnica, não deve haver nenhuma interferência manual do usuário em ambas as tecnologias, GPT-3 e GPT-4), tempo de resposta e organização estrutural. A Tabela 2 a seguir sintetiza os principais resultados obtidos após a realização do desafio:

Tabela 2 - Comparativo entre o desempenho do GPT-3 e do GPT-4 nos testes práticos

Critério	GPT-3	GPT-4
Clareza do código	Regular – uso de variáveis genéricas e ausência de comentários	Boa – variáveis descritivas e comentários explicativos
Precisão da execução	Parcial – blocos inseridos de forma irregular; nenhuma tocha foi posicionada ao lado dos blocos inseridos.	Alta – execução fluida e posicionamento correto das tochas, mas não respeitou os intervalos de adição, nem a quantidade de blocos
Necessidade de correções	Sim – pequenos ajustes foram necessários antes da execução	Não – código executado integralmente sem alterações
Tempo de geração	11 segundos.	17 segundos.

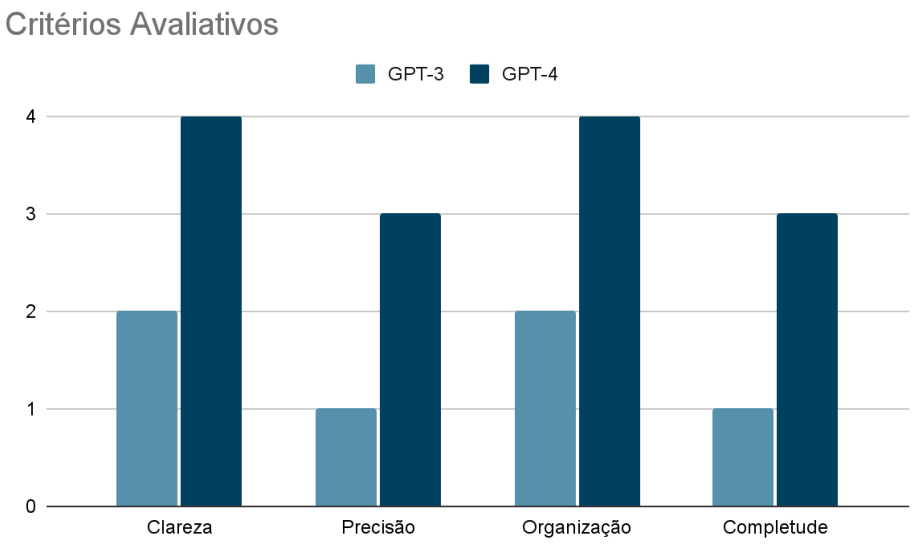
Organização e estrutura	Básica – código linear, sem modularização	Boa – uso de funções, indentação adequada e estrutura organizada

Fonte: Autor

A análise comparativa confirma uma evolução significativa no desempenho do GPT-4 em relação ao GPT-3. O novo modelo demonstrou maior capacidade de interpretação, estrutura de código mais clara e maior autonomia na entrega de soluções completas. Enquanto o GPT-3 ofereceu uma base funcional, porém limitada, o GPT-4 destacou-se pela maturidade técnica, evidenciando avanços concretos no suporte ao desenvolvimento de software.

Observa-se na Figura 7 que o GPT-4 supera o GPT-3 em clareza, precisão, organização e completude. Ainda que tenha alcançado 3/5 em completude — valor superior ao do GPT-3 — o modelo não concluiu integralmente o desafio conforme especificado; por protocolo, intervenções só ocorreriam se a resposta não executasse, e como ambas executaram, manteve-se a execução literal para fins comparativos.

Figura 7 - Comparação visual das notas por critério (1–5) entre GPT-3 e GPT-4



Fonte: Autor (2025)

O atendimento aos requisitos do desafio é sintetizado na Tabela 3, indicando de forma binária quais itens foram cumpridos por cada modelo.

Tabela 3 - Checklist dos requisitos do desafio por cada modelo

Requisito de Avaliação	GPT-3	GPT-4
Execução sem intervenção humana (reprodução literal)	x	x
Estrada linear concluída (20 blocos)		

Tochas a cada 5 blocos nas posições corretas		
Compilação/execução sem erro	x	x
Tempo de geração ≤ 30 s	x	x
Organização em função/método reutilizável		x
Comentários explicativos		x

Fonte: Autor (2025)

Conforme a Tabela 3, ambos executam sem intervenção e dentro do tempo, mas não concluem integralmente a estrada nem posicionam todas as tochas; o GPT-4 se destaca em organização e documentação do código.

Constata-se que a maior parte dos acertos está compartilhada entre os modelos (42,86%), com vantagem adicional do GPT-4 (28,57%) e duas exigências ainda não atendidas por nenhum (28,57%).

4.4. Experiência do programador

Além dos critérios técnicos, esta seção apresenta uma análise subjetiva da experiência do programador ao interagir com os modelos GPT-3 e GPT-4 durante os testes. São abordadas percepções relacionadas à clareza dos códigos, necessidade de ajustes, facilidade de leitura e impacto na produtividade.

Na prática profissional, a diferença entre um código mal estruturado (como o gerado pelo GPT-3) e um código mais organizado (como o do GPT-4) pode significar horas adicionais de manutenção, retrabalho ou até falhas em produção. Em projetos reais, especialmente em times de desenvolvimento, a clareza do código impacta diretamente a colaboração entre desenvolvedores. Um código sem comentários ou com lógica pouco intuitiva tende a dificultar revisões em pull requests[1], além de comprometer a escalabilidade de sistemas. Assim, as limitações observadas no exemplo do Minecraft ilustram situações reais em que o programador precisa adaptar ou corrigir respostas automáticas antes de integrar a solução em projetos de maior porte.

4.4.1. Facilidade de leitura e entendimento do código

Durante os testes, constatou-se uma diferença notável entre os códigos gerados pelos dois modelos. O GPT-4 apresentou uma estrutura mais clara, com variáveis nomeadas de

[1] Pull request: solicitação feita em sistema de versionamento (como github ou gitlab) para que um trecho de código seja revisado e integrado ao repositório principal do projeto.

forma intuitiva e presença de comentários explicativos, o que facilitou significativamente a leitura e o entendimento da lógica implementada.

Já o GPT-3, embora tenha funcionado, produziu um código mais direto, bem limpo e pouco documentado, com variáveis genéricas e ausência de comentários, exigindo maior esforço interpretativo por parte do programador.

4.4.2. Intervenções manuais necessárias

O código gerado pelo GPT-3 demandou pequenos ajustes manuais, como correções de indentação e inclusão de uma importação ausente. Além disso, foi necessário revisar a lógica de posicionamento das tochas, que estavam sendo colocadas apenas de um lado da estrada.

Em contrapartida, o código do GPT-4 foi executado integralmente sem necessidade de alterações, representando um ganho prático em termos de agilidade e confiabilidade.

4.4.3. Sensação de controle, confiança e clareza

A experiência com o GPT-4 proporcionou maior sensação de controle e confiança sobre o resultado. O modelo demonstrou melhor capacidade de interpretar comandos complexos e gerou soluções mais limpas e organizadas, o que aumentou a segurança do programador ao executar diretamente o código.

Já com o GPT-3, foi necessário manter atenção constante, revisar logicamente o que foi gerado e realizar validações adicionais antes da execução, o que compromete a fluidez do processo.

4.4.4. Considerações sobre produtividade

O GPT-4 se destacou em termos de produtividade, ao reduzir significativamente o tempo gasto com edições, testes e correções. Sua capacidade de gerar código pronto para uso contribuiu para a agilidade do processo e otimização do tempo do programador.

O GPT-3, embora útil, exigiu mais etapas de revisão e ajustes, o que representa uma limitação em contextos que demandam entregas rápidas ou fluxo contínuo de trabalho.

4.5. Impacto na produtividade e aprendizado

A aplicação prática dos modelos GPT-3 e GPT-4 evidenciou impactos significativos tanto na produtividade do programador quanto no processo de aprendizado. A proposta de automatizar uma tarefa simples no Minecraft, utilizando código em Python, permitiu

identificar diferenças marcantes entre os dois modelos, não apenas no aspecto técnico, mas também na experiência pedagógica proporcionada.

Do ponto de vista da produtividade, o GPT-4 apresentou desempenho superior ao gerar um código funcional, bem estruturado e pronto para execução imediata, sem necessidade de correções manuais. Essa agilidade se traduziu em economia de tempo e maior fluidez no processo. O GPT-3, por sua vez, embora funcional, exigiu edições e ajustes prévios à execução, o que retardou a conclusão da tarefa e aumentou a necessidade de intervenção humana.

Em relação ao aprendizado, o GPT-4 se mostrou mais eficaz como ferramenta de apoio educacional. O código gerado apresentava estrutura lógica clara, comentários explicativos e uso de boas práticas de programação, o que favoreceu a compreensão por parte do programador, especialmente iniciantes. O GPT-3, embora também funcional, entregou um código mais direto, porém com menor valor explicativo e didático.

Entretanto, o uso frequente de modelos como o GPT-4 também levanta um ponto de atenção: o risco de dependência cognitiva. A facilidade em obter soluções completas pode desestimular o aprendizado ativo e a prática autônoma de resolução de problemas. Como aponta Souza (2023), é fundamental que o uso dessas ferramentas seja orientado de forma crítica, de modo a complementar e não substituir o desenvolvimento de competências fundamentais na programação.

Em síntese, a experiência demonstrou que a IA pode atuar como uma aliada estratégica na produtividade e no aprendizado, desde que seu uso seja consciente. O GPT-4, em especial, apresenta maturidade técnica e potencial de integração em contextos educacionais e profissionais. Contudo, é imprescindível que o programador mantenha uma postura analítica, validando os resultados e participando ativamente do processo de desenvolvimento.

5. Considerações Éticas e Profissionais no Uso da IA

A adoção de modelos de linguagem como o GPT-3 e o GPT-4 no desenvolvimento de software tem promovido ganhos substanciais em termos de produtividade e eficiência. No entanto, à medida que essas ferramentas se tornam cada vez mais presentes no cotidiano profissional, surgem também desafios éticos e responsabilidades que extrapolam o aspecto técnico. O uso da inteligência artificial nesse contexto envolve decisões que impactam diretamente a prática da profissão, exigindo uma abordagem consciente.

5.1. Responsabilidade e Autoria

Um dos principais dilemas relacionados ao uso da IA é a atribuição de autoria. Quando uma ferramenta como o GPT-4 é capaz de gerar um trecho completo de código a partir de uma simples instrução, torna-se legítima a discussão sobre quem é, de fato, o autor daquele conteúdo. Floridi et al. (2018) destacam que, em uma sociedade moldada por sistemas inteligentes, é necessário repensar os limites da responsabilidade e da autoria, especialmente quando os humanos atuam como intermediários no processo de criação.

Em ambientes educacionais, essa discussão se intensifica. Estudantes que entregam códigos produzidos por IA sem compreender sua lógica interna correm o risco de comprometer sua formação e apresentar um desempenho ilusório. No mercado de trabalho, esse mesmo comportamento pode afetar a confiança entre membros da equipe, clientes e gestores, gerando dúvidas sobre a real competência técnica envolvida.

5.2. Riscos de dependência e superficialidade no aprendizado

A capacidade do GPT-4 de fornecer soluções completas e prontas para uso representa um avanço notável em termos de agilidade e eficiência. No entanto, essa vantagem pode gerar efeitos colaterais importantes, um dos principais riscos é a formação de uma dependência excessiva por parte do programador, o que pode comprometer o desenvolvimento de competências fundamentais, como raciocínio lógico, depuração manual de erros e construção autônoma de estruturas de código.

De acordo com Souza (2023), o uso recorrente de sistemas baseados em IA, quando realizado sem reflexão crítica, tende a favorecer um aprendizado superficial e mecanizado. Em contextos educacionais, isso pode comprometer a formação técnica do estudante; em ambientes profissionais, pode afetar a capacidade de adaptação a cenários imprevistos.

Diante disso, é essencial compreender a IA como uma ferramenta de apoio complementar, e não como substituta do pensamento crítico e da prática ativa. O domínio do conteúdo deve permanecer com o humano, enquanto a IA atua como um recurso de extensão e aprimoramento, e não como um atalho para soluções prontas.

5.3. Imprecisão e revisão crítica

Embora o GPT-4 represente um avanço considerável em relação ao GPT-3, os modelos de linguagem continuam suscetíveis a gerar respostas imprecisas, incompletas ou até mesmo incorretas. Isso se deve à forma como esses sistemas operam: como explicam Bender et al.

(2021), modelos como o GPT-4 não compreendem o conteúdo que produzem, eles apenas identificam e reproduzem padrões linguísticos com base em grandes volumes de dados utilizados no treinamento.

Dessa maneira, mesmo que uma resposta pareça coerente ou convincente, ela pode conter falhas lógicas, omissões relevantes ou vieses herdados dos dados de origem. Tais limitações são particularmente críticas em aplicações técnicas, como o desenvolvimento de software, onde a precisão e a segurança são fundamentais.

Por isso, é indispensável que o programador atue como revisor crítico das respostas geradas. Nenhum código deve ser utilizado sem uma análise cuidadosa quanto à sua funcionalidade, coerência e adequação ao contexto de uso. Independentemente da sofisticação da IA, a responsabilidade final sobre o resultado continua sendo humana.

5.4. Boas práticas profissionais no uso de IA

Diante dos desafios e responsabilidades discutidos, torna-se essencial adotar boas práticas que orientem o uso ético e consciente de ferramentas baseadas em inteligência artificial, como o ChatGPT, GitHub Copilot e Amazon CodeWhisperer. Essas práticas não apenas evitam riscos, mas também reforçam a atuação crítica e responsável do programador. Entre as principais recomendações, destacam-se:

- Revisar cuidadosamente todo o código gerado pela IA antes de sua implementação, assegurando correção e adequação ao contexto;
- Indicar claramente o uso da IA na construção de trechos de código, especialmente em ambientes acadêmicos ou de pesquisa, para garantir transparência e integridade;
- Evitar o uso indiscriminado de IA em tarefas sensíveis, principalmente aquelas que envolvem decisões críticas, segurança de dados ou informações confidenciais;
- Estimular o aprendizado ativo e a autonomia do programador, utilizando a IA como ferramenta complementar e não como substituta do processo de desenvolvimento intelectual.

A adoção dessas práticas fortalece o papel do profissional como agente crítico e consciente, contribuindo para um ambiente de desenvolvimento mais ético, seguro e sustentável.

5.5. Referenciais éticos da computação

À luz dos resultados desta pesquisa, que evidenciam ganhos de clareza, depuração e produtividade com modelos de linguagem (GPT-3 e GPT-4), ressalta-se que a adoção dessas ferramentas no desenvolvimento de software deve observar princípios profissionais de ética

em computação. O Código de Ética e Conduta Profissional da Sociedade Brasileira de Computação orienta para benefício social, honestidade intelectual, transparência e proteção de dados, diretrizes diretamente pertinentes ao uso de IA em programação, especialmente quando há geração automática de trechos de código e apoio a decisões técnicas. Nesse sentido, recomenda-se explicitar o uso de IA nos artefatos do projeto, revisar e testar o código gerado e evitar o envio de informações sensíveis a serviços de terceiros. (SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2024).

Em síntese, a incorporação de IA deve harmonizar ganhos de desempenho com responsabilidade profissional: documentar versões/modelos empregados, validar resultados e preservar privacidade e segurança estão em consonância com o referido Código e fortalecem a qualidade e a confiabilidade dos produtos de software derivados deste trabalho. (SOCIEDADE BRASILEIRA DE COMPUTAÇÃO, 2024).

5.6. Análise crítica por contexto de uso (GPT-3 e GPT-4)

Embora os resultados indiquem desempenho superior do GPT-4 em precisão, clareza e completude, a adoção do modelo deve considerar contexto, custo e latência. Evidências externas também apontam ganhos do GPT-4 em tarefas que exigem raciocínio mais elaborado e explicações mais consistentes (OPENAI, 2023). Em contrapartida, há cenários em que o GPT-3 permanece adequado e eficiente, especialmente quando há baixa complexidade e restrições de orçamento/tempo.

- **Quando priorizar o GPT-4 (recomendado):** problemas com múltiplas etapas ou dependências entre módulos; depuração de erros não triviais; geração de testes explicados; necessidade de justificativas mais claras e menor ambiguidade (OPENAI, 2023).
- **Quando o GPT-3 é suficiente/útil:** tarefas rotineiras e de baixa complexidade (snippets, CRUD simples, regex, refatorações localizadas), brainstorming e rascunhos de documentação, ou processamento em alto volume com latência/custo mais restritos, desde que se mantenha a revisão humana e testes antes da integração.

Em síntese, a escolha entre modelos deve ser guiada pelo risco e pela complexidade da tarefa: quanto maior a exigência de confiabilidade e explicações, mais justificável o uso do GPT-4; quanto mais operacionais e repetitivas forem as demandas, mais viável o GPT-3.

6. Considerações Finais

Este estudo evidenciou de forma consistente a evolução no suporte proporcionado por modelos de linguagem baseados em inteligência artificial, ao comparar o desempenho do GPT-3 e do GPT-4 no contexto do desenvolvimento de software. Por meio de uma tarefa prática implementada no ambiente do jogo Minecraft, foi possível observar diferenças marcantes entre os dois modelos, especialmente em termos de clareza, organização, precisão e autonomia na geração de código.

O GPT-4 destacou-se por sua maior capacidade de interpretação contextual, pela produção de código estruturado e legível, e pela ausência de necessidade de correções manuais. Tais características impactaram diretamente a produtividade e a confiança do programador na aplicação imediata do código. Em contrapartida, o GPT-3, embora funcional, apresentou limitações significativas na interpretação de instruções compostas e exigiu intervenções antes da execução.

Sob a perspectiva educacional, o GPT-4 demonstrou maior eficácia como ferramenta de apoio ao aprendizado, oferecendo explicações e estruturas que favorecem a compreensão de boas práticas de programação. No entanto, o uso de modelos de IA como esse deve ser acompanhado de reflexão crítica, a fim de evitar a dependência excessiva.

Conclui-se que o GPT-4 representa um avanço significativo no suporte ao desenvolvimento de software, oferecendo maior clareza, precisão e autonomia em comparação ao GPT-3. No entanto, seu uso deve ser pautado por responsabilidade e revisão crítica, já que não substitui o papel do programador. Enquanto o GPT-4 é mais indicado para tarefas complexas e que exigem explicações detalhadas, o GPT-3 permanece uma alternativa viável em atividades rotineiras e de menor complexidade, especialmente quando há restrições de custo e tempo.

Diferentemente de trabalhos que se limitam a relatos gerais, este estudo executou uma tarefa prática replicável e comparou, lado a lado, as saídas dos modelos, usando critérios operacionais (clareza, precisão, completude e necessidade de intervenção humana). Esse desenho permitiu demonstrar, de forma objetiva, onde o GPT-4 entrega ganhos concretos (interpretação contextual e código mais utilizável) e, ao mesmo tempo, registrar em quais cenários o GPT-3 permanece adequado (tarefas simples, baixo risco e restrições de custo/latência), oferecendo um guia de decisão por contexto, contribuição aplicada que nem sempre é explicitada em estudos anteriores.

Para empresas de software, este trabalho oferece um critério prático de escolha: o GPT-3 permanece uma opção eficiente e econômica para tarefas rotineiras e de baixo risco, entregando bons resultados com agilidade; o GPT-4 se mostra mais indicado quando a demanda é complexa, exige maior precisão, entendimento de contexto, várias etapas encadeadas e respostas menos ambíguas. Em síntese, adota-se GPT-3 como padrão para o cotidiano e eleva-se para GPT-4 quando complexidade, confiabilidade e clareza se tornam críticas, ponderando custo e latência.

Além disso, a pesquisa apresenta uma contribuição prática relevante tanto para empresas de software quanto para o ensino de programação. Para as organizações, o comparativo entre os modelos oferece subsídios para a escolha consciente da tecnologia mais adequada às suas demandas, equilibrando custo, confiabilidade e produtividade. No campo educacional, os resultados servem como referência para professores e estudantes, apoiando metodologias de ensino que integrem a IA de forma crítica e responsável no processo de aprendizagem da programação.

7. Trabalhos Futuros

Para pesquisas futuras, propõe-se aplicar a mesma metodologia com o GPT-5, utilizando os mesmos critérios e tarefas. Dessa forma, será possível avaliar se a evolução observada do GPT-3 para o GPT-4 se mantém e se o GPT-5 traz ganhos significativos na qualidade, clareza e autonomia das soluções geradas comparado ao seu antecessor, que é o GPT-4.

8. Referências

AMAZON WEB SERVICES. Amazon CodeWhisperer — Documentation. 2023. Disponível em: https://docs.aws.amazon.com/codewhisperer/. Acesso em: 05 mai. 2025.

BENDER, E. M.; GEBRU, T.; McMILLAN-MAJOR, A.; SHMITCHELL, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In: Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency (FAccT '21). New York: ACM, 2021. p. 610–623. DOI: 10.1145/3442188.3445922. Disponível em: https://doi.org/10.1145/3442188.3445922. Acesso em: 12 mai. 2025.

BROWN, T. B. et al. Language Models are Few-Shot Learners. *arXiv preprint*, arXiv:2005.14165, 2020. Disponível em: https://arxiv.org/abs/2005.14165. Acesso em: 20 mai. 2025.

FERRETI, J.; SALESI, V.; CAVICHIOLLI, A. *Inteligência Artificial: uma jornada de descobertas e aprendizados*. Três Corações: Universidade Vale do Rio Verde (UninCor), 2024. Disponível em: https://repositorio.unincor.edu.br/items/2dfa650a-58a0-46e9-9cd7-f08c46a60c2f. Acesso em: 28 mai. 2025.

FLORIDI, L. et al. An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations. *Minds and Machines*, v. 28, n. 4, p. 689–707, 2018. Disponível em: https://doi.org/10.1007/s11023-018-9482-5. Acesso em: 04 jun. 2025.

GITHUB. Introducing GitHub Copilot: your AI pair programmer. *GitHub Blog*, 29 jun. 2021. Disponível em: https://github.blog/news-insights/product-news/introducing-github-copilot-ai-pair-programmer/. Acesso em: 04 jun. 2025.

JURAFSKY, D.; MARTIN, J. H. Speech and Language Processing. 3rd ed. (draft). 2024. Disponível em: https://web.stanford.edu/~jurafsky/slp3/. Acesso em: 15 jun. 2025.

LOSIO, R. Amazon CodeWhisperer Now Generally Available. InfoQ, 13 abr. 2023. Disponível em: https://www.infoq.com/news/2023/04/amazon-codewhisperer-generally/. Acesso em: 22 jun. 2025.

NASH, L. GitHub Copilot is now generally available to individual developers. GitHub Blog, 21 jun. 2022. Disponível em: https://github.blog/news-insights/product-news/github-copilot-is-now-generally-available-to-individual-developers/. Acesso em: 01 jul. 2025.

OPENAI. GPT-4 Technical Report. arXiv preprint, arXiv:2303.08774, 2023. Disponível em: https://arxiv.org/abs/2303.08774. Acesso em: 10 jul. 2025.

POLD RACK, R. A. AI-assisted coding: Experiments with GPT-4. arXiv preprint, arXiv:2304.13187, 2023. Disponível em: https://arxiv.org/abs/2304.13187. Acesso em: 18 jul. 2025.

REPLIT. Building Ghostwriter — How Replit built an AI pair programmer. Replit Blog, 2023. Disponível em: https://blog.replit.com/building-ghostwriter. Acesso em: 25 jul. 2025.

SOUZA, D. L. L. de. Estudo de caso: uso do ChatGPT para resolução de problemas de programação. 2023. Trabalho de Conclusão de Curso (Graduação em Sistemas de Informação) — Universidade Federal de Campina Grande. Disponível em: https://repositorio.ufcg.edu.br/jspui/handle/20.500.11928/12345. Acesso em: 30 jul. 2025.

SOCIEDADE BRASILEIRA DE COMPUTAÇÃO (SBC). *Código de Ética e Conduta Profissional*. Resolução nº 02, 21 mar. 2024. Disponível em: https://www.sbc.org.br/etica-sbc/. Acesso em: 15 jun. 2025..

WIKIMEDIA COMMONS. Ciclo de vida do desenvolvimento de software. 2023. Disponível em: https://commons.wikimedia.org/wiki/File:SDLC_-_Software_Development_Life_Cycle.jpg . Acesso em: 30 set. 2025.