



UNIVERSIDADE FEDERAL DO MARANHÃO
Graduação em Engenharia da Computação

Ednildo Cunha Almeida

**Avanços na Resolução de Problemas de
Programação: Uma Comparação entre *Chat
GPT 3.5* e *Chat GPT 4* através de Desafios da
BeeCrowd**

São Luís

2024

Ednildo Cunha Almeida

**Avanços na Resolução de Problemas de Programação:
Uma Comparação entre *Chat GPT 3.5* e *Chat GPT 4*
através de Desafios da BeeCrowd**

Trabalho de Conclusão de Curso II
apresentado ao curso de Engenharia da
Computação da Universidade Federal
do Maranhão, como parte dos requisitos
necessários para obtenção do grau de
Bacharel em Engenharia da Computação.

Orientador: Prof. Dr. Paulo Rogério de Almeida Ribeiro

São Luís

2024

Ficha gerada por meio do SIGAA/Biblioteca com dados fornecidos pelo(a) autor(a).
Diretoria Integrada de Bibliotecas/UFMA

Almeida, Ednildo Cunha.

Avanços na resolução de problemas de programação : uma comparação entre Chat GPT 3.5 e Chat GPT 4 através de desafios da BeeCrowd / Ednildo Cunha Almeida. - 2024.
50 f.

Orientador(a): Paulo Rogerio de Almeida Ribeiro.
Curso de Engenharia da Computação, Universidade Federal do Maranhão, São Luís, Maranhão, 2024.

1. Análise Comparativa. 2. Chat Gpt 3.5. 3. Chat Gpt 4. 4. Plataforma Beecrowd. 5. . I. Ribeiro, Paulo Rogerio de Almeida. II. Título.

Ednildo Cunha Almeida

**Avanços na Resolução de Problemas de Programação:
Uma Comparação entre *Chat GPT 3.5* e *Chat GPT 4*
através de Desafios da BeeCrowd**

Trabalho de Conclusão de Curso II
apresentado ao curso de Engenharia da
Computação da Universidade Federal
do Maranhão, como parte dos requisitos
necessários para obtenção do grau de
Bacharel em Engenharia da Computação.

**Prof. Dr. Paulo Rogério de Almeida
Ribeiro**
Orientador

Prof. Dr. Alex Oliveira Barradas Filho
Examinador

Prof. Dr. Bruno Feres de Souza
Examinador

São Luís
2024

Agradecimentos

Primeiramente, gostaria de oferecer este trabalho em memória de minha sogra, Maria das Dores Silva Damasceno, que tenho certeza que estaria imensamente feliz por me ver obtendo esta conquista. De onde estiver sei que está intercedendo pela nossa família e guiando-nos com sua luz.

Agradeço profundamente à minha mãe, Maria Amélia Cunha Almeida, e à minha avó, Raimunda Nonata Cunha Almeida, que sempre foram pilares de força e sabedoria na minha vida. Vocês me mostraram desde cedo que a educação é o único caminho a seguir, e este trabalho é fruto desse aprendizado que me passaram com tanto amor e perseverança.

Gostaria também de expressar meus agradecimentos, à minha namorada Karla Cristina Silva Barros, que esteve ao meu lado durante esta jornada, me apoiando e incentivando a continuar, mesmo nas horas mais difíceis que passamos. Sua presença foi essencial para que eu pudesse concluir este trabalho. Obrigado por todo o apoio e carinho.

Por último, gostaria de expressar minha imensa gratidão ao meu orientador, Prof. Dr. Paulo Rogério de Almeida Ribeiro. Seus incentivos, sua paciência e, claro, os necessários puxões de orelha, foram fundamentais não apenas para minha formação como profissional, mas também para meu crescimento como pessoa. Obrigado por sempre acreditar no meu potencial e por me guiar com tanta dedicação.

A todos, meu sincero agradecimento.

*"Nada na vida deve ser temido, somente compreendido.
Agora é hora de compreender mais para temer menos."*

Marie Curie

Resumo

O presente trabalho apresenta uma análise comparativa entre os modelos de linguagem natural Chat GPT 3.5 e Chat GPT 4, focando na resolução de problemas de programação na plataforma BeeCrowd. O objetivo central é avaliar a eficácia das duas versões do modelo na resolução de 50 problemas categorizados em níveis de dificuldade (fácil, intermediário e difícil), replicando um experimento realizado anteriormente com o Chat GPT 3.5.

A pesquisa utiliza uma metodologia de submissão de problemas à plataforma, comparando os resultados obtidos pelas duas versões do Chat GPT. O estudo abrange aspectos como a taxa de acertos e erros, a quantidade de tentativas necessárias para a resolução correta e tipos de erros encontrados. Além disso, o trabalho explora as limitações do Chat GPT 4, especialmente na interpretação de exemplos de entrada e saída nas questões de programação, o que resultou em um desempenho inferior ao do Chat GPT 3.5.

Os resultados mostraram que o Chat GPT 3.5 obteve uma taxa de acerto de 42%, enquanto o Chat GPT 4 alcançou apenas 2% de sucesso. A pesquisa também levanta hipóteses sobre o baixo desempenho do Chat GPT 4, bem como realiza experimentos adicionais para discutir tais hipóteses, destacando a importância de fornecer instruções explícitas sobre os exemplos nas questões para melhorar a precisão das respostas geradas.

Conclui-se que, apesar dos avanços tecnológicos incorporados no Chat GPT 4, seu desempenho na resolução de problemas de programação não superou o de seu antecessor.

Palavras-chave: Análise Comparativa, Chat GPT 3.5, Chat GPT 4, plataforma BeeCrowd.

Abstract

The present work presents a comparative analysis between the natural language models Chat GPT 3.5 and Chat GPT 4, focusing on solving programming problems on the BeeCrowd platform. The main objective is to evaluate the effectiveness of the two versions of the model in solving 50 problems categorized by difficulty levels (easy, intermediate, and hard), replicating a previously conducted experiment with Chat GPT 3.5.

The research employs a problem submission methodology to the platform, comparing the results obtained by both versions of Chat GPT. The study covers aspects such as success and failure rates, the number of attempts required for correct resolution, and types of errors encountered. Additionally, the work explores the limitations of Chat GPT 4, particularly in interpreting input and output examples in programming problems, which resulted in a lower performance compared to Chat GPT 3.5.

The results showed that Chat GPT 3.5 achieved a success rate of 42%, while Chat GPT 4 only reached a 2% success rate. The research also raises hypotheses about Chat GPT 4's low performance and conducts additional experiments to discuss these hypotheses, highlighting the importance of providing explicit instructions regarding examples in the questions to improve the accuracy of the generated responses.

It concludes that, despite the technological advances incorporated into Chat GPT 4, its performance in solving programming problems did not surpass that of its predecessor.

Keywords: Comparative Analysis, Chat GPT 3.5, Chat GPT 4, BeeCrowd Platform.

Lista de ilustrações

Figura 1 – Questão BeeCrowd	25
Figura 2 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 3.5</i>	29
Figura 3 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i>	29
Figura 4 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 3.5</i>	30
Figura 5 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i>	30
Figura 6 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 3.5</i>	32
Figura 7 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i>	32
Figura 8 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com <i>Chat GPT 3.5</i>	33
Figura 9 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com <i>Chat GPT 4</i>	33
Figura 10 – Exemplo de código gerado com <i>Chat GPT 4</i>	34
Figura 11 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i> sem indicar o exemplo.	35
Figura 12 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	35
Figura 13 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i> sem indicar o exemplo.	36
Figura 14 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	36
Figura 15 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i> sem indicar o exemplo.	37
Figura 16 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	37
Figura 17 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com <i>Chat GPT 4</i> sem indicar o exemplo.	38
Figura 18 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	38

Figura 19 – Gráfico com proporção dos 50 problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	40
Figura 20 – Gráfico com a proporção de acertos e erros por grau de dificuldades dos 50 problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	40
Figura 21 – Gráfico com número de acertos por número de tentativas e grau de dificuldades dos 50 problemas respondidos na plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	41
Figura 22 – Gráfico com os erros gerados pelas respostas dos 50 problemas respondidos submetidas à plataforma BeeCrowd com <i>Chat GPT 4</i> indicando o exemplo.	42

Lista de abreviaturas e siglas

IA	<i>Inteligência Artificial</i>
GPT	<i>Generative Pre-Trained Transformer(Transformador Pré-treinado Generativo)</i>
PLN	<i>Processamento de Linguagem Natural</i>
MIT	<i>Massachusetts Institute of Technology</i>

Sumário

1	INTRODUÇÃO	13
1.1	Introdução aos Modelos de Processamento de Linguagem Natural (PLN)	13
1.2	Impacto do <i>Chat GPT</i> na sociedade	13
1.3	<i>Chat GPT</i> na Programação	14
1.4	Necessidade de Avaliação Empírica	16
1.5	Estudo Original de Souza (2023)	17
1.5.1	Descrição do Projeto Base	17
1.5.2	Resultados e Discussão do Projeto Base	17
1.6	Objetivos	19
1.6.1	Objetivo Geral	19
1.6.2	Objetivos Específicos	19
2	FUNDAMENTAÇÃO TEÓRICA	20
2.1	Histórico e Evolução	20
2.2	<i>Chat GPT</i>	21
2.3	Arquitetura Transformer	22
2.3.1	Componentes Básicos da Arquitetura	22
2.3.2	Mecanismo de Atenção	22
2.3.2.1	Tipos de Atenção	23
2.3.3	Codificação Posicional (<i>Positional Encoding</i>)	23
2.3.4	Camadas de <i>Feed-Forward</i>	23
2.3.5	Normalização e Conexões Residuais	24
2.3.6	Capacidades de Pré-Treinamento e Ajuste Fino	24
3	METODOLOGIA	25
3.1	Seleção dos Problemas	25
3.2	Uso do <i>Chat GPT 4</i>	25
3.3	Procedimento de Submissão	26
3.4	Limitação de Interações	26
3.5	Documentação e Análise dos Resultados	26
3.6	Avaliação e Comparação dos Resultados	26
3.7	Hipóteses e Experimentos Complementares	26
3.7.1	Análise dos Resultados dos Experimentos Complementares	27
3.8	Resultados Esperados	27

4	RESULTADOS E DISCUSSÃO	28
4.1	<i>Chat GPT 4</i> em comparação ao <i>Chat GPT 3.5</i>	28
4.2	Hipótese levantada para o baixo desempenho do <i>Chat GPT 4</i>	33
4.3	Comparação entre <i>Chat GPT 3.5</i> e <i>Chat GPT 4</i> com o texto indicando os exemplos	39
5	CONCLUSÃO	43
5.1	Trabalhos Futuros	44
	REFERÊNCIAS	45
	ANEXOS	47
	ANEXO A – EXEMPLO DE QUESTÃO PROPOSTA AO <i>CHAT</i> <i>GPT4</i>	48
	ANEXO B – EXEMPLO DE CÓDIGO RESPOSTA DO <i>CHAT GPT4</i>	50

1 Introdução

A evolução da Inteligência Artificial (IA) tem revolucionado várias áreas do conhecimento. Entre as ferramentas emergentes, o *Chat GPT*, desenvolvido pela *OpenAI*, destaca-se por seu impacto significativo em diferentes áreas da sociedade.

Este trabalho visa explorar o efeito do *Chat GPT* no campo da programação, através de uma análise empírica que investiga a utilização da ferramenta na resolução de problemas apresentados em plataformas de desafios de código.

1.1 Introdução aos Modelos de Processamento de Linguagem Natural (PLN)

Para chegar à evolução apresentada nos modelos mais utilizados como o *Chat GPT*, tivemos um longo estudo sobre Processamento de Linguagem Natural (PLN).

Hai (2023) descreve o PLN como uma subárea da inteligência artificial dedicada a permitir que computadores entendam e gerem linguagem humana.

Ele cita que inicialmente, os modelos de PLN utilizavam regras específicas, mas essas abordagens não eram eficazes ao lidar com a complexidade e ambiguidade da linguagem natural. Com o avanço das redes neurais profundas e de arquiteturas como o *Transformer*, o PLN evoluiu significativamente, permitindo que os sistemas aprendessem automaticamente a partir de grandes volumes de dados e melhorassem o processamento e a geração de texto.

Por fim o autor mostra que modelos como o GPT, por exemplo, tornaram-se populares por sua capacidade de gerar textos de alta qualidade e estão sendo aplicados em áreas como atendimento ao cliente, tradução automática e geração de conteúdo.

Adicionalmente, observa-se que apesar desses avanços, o PLN ainda enfrenta desafios importantes, como a privacidade e a segurança das informações dos usuários, além da necessidade de aumentar a precisão em contextos mais complexos. O futuro do PLN depende da contínua evolução dessas tecnologias, juntamente com a implementação de práticas éticas e seguras no desenvolvimento desses modelos (HAI, 2023).

1.2 Impacto do *Chat GPT* na sociedade

O artigo, "Impacto do *Chat GPT* na Sociedade", escrito por Soares (2023), examina as principais implicações da tecnologia de IA *Chat GPT* em diferentes aspectos da sociedade. Ele começa com uma introdução sobre o que é o *Chat GPT*, um modelo de inteligência

artificial desenvolvido pela *OpenAI* que utiliza a tecnologia *GPT* (*Generative Pre-trained Transformer*) para gerar textos e responder a perguntas.

O autor continua falando sobre os impactos no mundo empresarial, onde o *Chat GPT* tem sido amplamente adotado para otimizar processos, melhorar a comunicação interna, aumentar a produtividade e auxiliar na tomada de decisões. Exemplos incluem sua utilização para automatizar tarefas repetitivas, como agendar reuniões, e para ajudar na criação de códigos de programação, com empresas relatando uma melhora na eficiência das operações.

Também sobre o campo da educação, o artigo discute os potenciais benefícios e desafios do *Chat GPT*. Enquanto ele pode ser uma ferramenta valiosa para apoiar o aprendizado, há preocupações sobre o possível impacto negativo na capacidade dos estudantes de desenvolver habilidades críticas e de pesquisa, caso haja uma dependência excessiva dessa tecnologia.

E conclui com uma análise das vantagens e desvantagens gerais do *Chat GPT*, destacando a rapidez com que ele fornece respostas e sua flexibilidade em várias áreas, ao mesmo tempo que alerta para os riscos, como a falta de compreensão de contexto e nuances, e possíveis questões de privacidade e segurança de dados.

Diante disso, surge o questionamento: "Como o uso do *Chat GPT* pode impactar na Programação?"

1.3 *Chat GPT* na Programação

A Programação é uma área em constante evolução, com isso o uso de tecnologias recentes como o *Chat GPT* para auxiliar vem sendo explorado desde as bases da educação até o âmbito profissional.

No artigo de Silva (2024) ele fala sobre o uso do *Chat GPT* na educação e seus principais desafios.

O autor se atém especialmente no ensino de programação. O estudo foca em como essa tecnologia pode ser integrada nas salas de aula, auxiliando tanto alunos quanto professores com atividades que vão desde a geração de trechos de código até a explicação de conceitos complexos. O *Chat GPT* também contribui para a sustentabilidade no ensino, permitindo aprendizado personalizado e adaptado ao ritmo de cada aluno.

Apesar dos benefícios, o artigo destaca desafios significativos, como a possível dependência dos estudantes em relação ao código gerado pela IA, o que pode prejudicar o desenvolvimento de habilidades críticas e a compreensão dos conceitos de programação. Também são levantadas preocupações éticas, como o plágio e o uso inadequado da

ferramenta para realizar tarefas avaliativas, o que poderia comprometer a integridade acadêmica.

A pesquisa envolveu a aplicação de questionários *Likert* a 40 estudantes brasileiros, com o objetivo de avaliar suas percepções sobre o uso do *Chat GPT* em aulas de programação. Os resultados mostraram que a maioria dos alunos reconhece o *Chat GPT* como uma ferramenta útil e inovadora, mas há uma preocupação significativa com o equilíbrio entre o uso da IA e o aprendizado autônomo. O artigo conclui que, embora o *Chat GPT* tenha potencial para transformar a educação em programação, é essencial que seu uso seja feito de forma equilibrada e responsável, a fim de promover uma compreensão profunda e sustentável da disciplina.

Já no meio profissional podemos ver o uso da ferramenta em diversas áreas como na criação de aplicações web, reduzindo o tempo necessário para concluir projetos e aumentando a eficiência dos programadores em obter respostas rápidas para dúvidas e sugestões de código, agilizando o processo de criação de software (MOTTA; ALVES; ANDRADE, 2023).

Porém apesar de a ferramenta oferecer vantagens, como a automatização de tarefas repetitivas e a geração de protótipos rápidos, também existem desafios, como a confiabilidade das respostas fornecidas e o risco de os programadores terceirizarem seu aprendizado para a IA, o que pode prejudicar o desenvolvimento de habilidades críticas (MOTTA; ALVES; ANDRADE, 2023).

Por fim conclui que, apesar das limitações e preocupações, o *Chat GPT* tem o potencial de impactar significativamente o desenvolvimento web, desde que utilizado de forma responsável e ética.

Outro exemplo de uso do modelo no âmbito profissional da programação é na criação de jogos.

Como demonstrado por Silva et al. (2023), em seu trabalho "Desenvolvimento de Jogos 2D de Plataforma: Explorando *Unity* e *Chat GPT* para Criação de Códigos Dinâmicos", que aborda o uso da plataforma *Unity*, uma *engine* popular para desenvolvimento de jogos, em conjunto com o *Chat GPT*. Demonstrando com essa parceria que os desenvolvedores podem otimizar o fluxo de trabalho, desde a codificação até a implementação de elementos de interface e lógica de jogo.

O *Chat GPT* pode também atuar em tarefas cotidianas na vida de desenvolvedores como na resolução de *bugs*. Explorando seu uso para ajudar a identificar e corrigir *bugs* em códigos de programação (SURAMEERY; SHAKOR, 2023).

O autor discute as capacidades do *Chat GPT*, como processamento de linguagem natural, reconhecimento de padrões e correção de erros, e como essas características podem

ser aproveitadas para fornecer assistência na depuração, previsão de *bugs* e explicação de erros.

Por fim, de modo geral temos que o *Chat GPT* pode ser um excelente aliado no ato de programar. O Trabalho de Conclusão de Curso "*How to Use Chat GPT for Programming*" explora como o *Chat GPT* pode ser utilizado para auxiliar na programação e desenvolvimento de uma aplicação móvel.

Durante o desenvolvimento do aplicativo, Seikkinen (2023) utilizou o *Chat GPT* para gerar trechos de código, sugerir ideias de design e ajudar na resolução de problemas relacionados ao código. A aplicação foi construída utilizando *React Native*, *TypeScript* e *Firebase* para autenticação e armazenamento de dados. Embora o *Chat GPT* tenha sido útil para acelerar o desenvolvimento, houve situações em que a IA não conseguiu fornecer soluções completas, o que exigiu conhecimento técnico adicional e pesquisa independente.

O artigo conclui que o *Chat GPT* pode melhorar o fluxo de trabalho dos desenvolvedores, mas não substitui a necessidade de conhecimento técnico. Ele serve como uma ferramenta de suporte, fornecendo sugestões e automação para tarefas repetitivas, mas ainda não tem a capacidade de projetar software de forma abrangente ou resolver problemas complexos sem intervenção humana.

1.4 Necessidade de Avaliação Empírica

Como visto o uso do *Chat GPT* na programação tem sido amplamente discutido, tanto no contexto educacional quanto profissional, devido ao seu potencial para transformar o modo como desenvolvedores e estudantes interagem com o código. No entanto, apesar dos benefícios teóricos e práticos que essa ferramenta de IA pode oferecer, ainda existem diversas lacunas no entendimento sobre seu impacto real no desenvolvimento de habilidades e na produção de software de qualidade. Essas lacunas ressaltam a necessidade de uma avaliação empírica mais robusta sobre o uso do modelo na programação.

Em suma, embora o *Chat GPT* apresente um grande potencial para auxiliar na programação, sua implementação em contextos educacionais e profissionais exige uma análise mais profunda e baseada em evidências empíricas. Somente com uma avaliação contínua e detalhada será possível determinar as melhores práticas para a integração dessa tecnologia, assegurando que ela seja utilizada de forma ética, responsável e verdadeiramente benéfica para o desenvolvimento de software e o aprendizado de programação.

1.5 Estudo Original de Souza (2023)

Esta investigação se inspira no artigo de DÉBORA LÊDA DE LUCENA SOUZA, que conduziu um estudo de caso focado na aplicação do *Chat GPT* para resolver desafios de programação.

1.5.1 Descrição do Projeto Base

No decorrer de sua pesquisa, Souza selecionou 100 questões de duas renomadas plataformas de programação, *BeeCrowd* e *LeetCode*, designando igualmente 50 problemas de cada plataforma. Estes desafios foram categorizados segundo seu grau de dificuldade: fácil, intermediário e difícil.

Para abordar esses desafios, Souza (2023) empregou a versão 3.5 do *Chat GPT*, copiando e colando na interface do chat o texto integral de cada problema contendo descrição, exemplos, e restrições, excluindo qualquer elemento gráfico associado. Em resposta, o *Chat GPT* fornecia soluções que eram, então, testadas na respectiva plataforma, sendo aceitas ou rejeitadas com base em sua correteza. Souza optou por ignorar quaisquer imagens vinculadas às questões, focando exclusivamente no texto.

Reconhecendo a potencial variabilidade nas respostas do *Chat GPT*, foi concedido ao modelo até três oportunidades de resolver cada problema. Caso as soluções propostas continuassem incorretas após três tentativas, o problema era classificado como não resolvido. Cada desafio foi processado individualmente em um *prompt* exclusivo, reutilizando-se o mesmo apenas para submissões subsequentes necessárias.

Os dados referentes a cada tentativa de solução, incluindo o enunciado do problema, sua dificuldade, o código da solução proposta pelo *Chat GPT*, o resultado da submissão (aceito ou rejeitado), e os erros reportados pela plataforma em caso de falha foram cuidadosamente registrados em uma planilha para análises futuras. A iteração do *Chat GPT* empregada no estudo foi atualizada em janeiro de 2023, baseando-se em dados compilados até setembro de 2021, assegurando respostas precisas e contemporâneas.

1.5.2 Resultados e Discussão do Projeto Base

Souza (2023) em seu estudo de caso sobre o uso do *Chat GPT* para resolução de problemas de programação, obteve resultados significativos ao avaliar o desempenho do modelo *Chat GPT* 3.5. O estudo incluiu a submissão de 100 problemas de programação selecionados aleatoriamente de plataformas populares como *LeetCode* e *BeeCrowd*, distribuídos entre os níveis de complexidade Fácil, Intermediário e Difícil. Para a plataforma *LeetCode*, os problemas foram divididos em 20 de nível fácil, 15 de nível intermediário e 15 de nível difícil. O estudo obteve os seguintes resultados:

- Todos os problemas foram resolvidos corretamente;
- 88% dos problemas foram resolvidos na primeira tentativa;
- Problemas que não foram resolvidos na primeira tentativa foram solucionados na segunda ou terceira tentativa.

Já para a plataforma *BeeCrowd*, que também teve seus 50 problemas que também foram distribuídos conforme a plataforma anterior, foram apresentados os seguintes resultados:

- Taxa de Acerto: 42% (21 problemas foram resolvidos corretamente);
- 36% dos problemas foram resolvidos na primeira tentativa;
- Problemas que não foram resolvidos na primeira tentativa foram submetidos novamente, com alguns sendo resolvidos na segunda ou terceira tentativa.

Os erros gerados durante a resolução dos problemas foram analisados, e identificou-se quatro tipos principais de erros:

- *Time Limit Exceeded*: Erro gerado quando a solução leva mais tempo do que o permitido para executar todos os testes de avaliação;
- *Wrong Answer (100%)*: Erro gerado quando a solução não apresenta o resultado esperado para todos os casos de teste.
- *Runtime Error*: Erro gerado ao definir um vetor ou array com capacidade inferior à necessária, ou ao tentar acessar uma posição inválida na memória.
- *Memory Limit Exceeded*: Erro gerado quando o código tenta alocar mais memória do que o máximo permitido para o problema.

O estudo de Souza (2023) concluiu que o *Chat GPT* pode ser utilizado para resolver uma gama de problemas de programação, especialmente de complexidade baixa a média. No entanto, seu uso requer atenção e revisão por parte dos programadores, já que nem sempre o resultado gerado estará correto. O desempenho insatisfatório em problemas de alta complexidade pode estar relacionado à estrutura mais elaborada dos enunciados e à inclusão de histórias fictícias e exemplos lúdicos.

Esses resultados forneceram uma base para as comparações propostas nesta pesquisa, destacando as áreas onde o modelo anterior teve sucesso e onde enfrentou dificuldades em comparação ao *Chat GPT 4*.

1.6 Objetivos

Avaliar a eficácia do *Chat GPT 4* na resolução de problemas de programação propostos pela plataforma *BeeCrowd*, comparando os resultados obtidos com aqueles do estudo anterior realizado por Souza (2023) com a versão 3.5 do *Chat GPT*. Este estudo visa contribuir para o entendimento de como as atualizações e melhorias na tecnologia da ferramenta *Chat GPT* podem impactar a capacidade de solucionar desafios de programação complexos.

1.6.1 Objetivo Geral

Realizar a comparação de desempenho entre o *Chat GPT 3.5* e o *Chat GPT 4* na resolução de problemas da plataforma *BeeCrowd*.

1.6.2 Objetivos Específicos

Destacam-se como objetivos específicos deste trabalho:

- Reproduzir o experimento anterior utilizando o *Chat GPT 4* para submeter os mesmos 50 problemas de programação da plataforma *BeeCrowd* escolhidos por Souza (2023). Esta reprodução visa garantir a consistência dos resultados e permitir uma comparação direta entre as versões 3.5 e 4 do *Chat GPT*.
- Empregar a versão mais atualizada do *Chat GPT*, versão 4, para resolver os problemas de programação, aproveitando sua base de treinamento ampliada e atualizada para verificar melhorias na resolução dos problemas.
- Comparar quantitativa e qualitativamente os resultados obtidos com o *Chat GPT 4* com aqueles alcançados no estudo de Souza (2023) usando o *Chat GPT 3.5*, com o intuito de identificar avanços significativos na precisão e eficiência da ferramenta de IA na resolução de problemas de programação.
- Investigar hipóteses adicionais por meio da realização de experimentos complementares, visando obter uma visão mais abrangente sobre os resultados encontrados durante a realização do estudo.

2 Fundamentação Teórica

2.1 Histórico e Evolução

A história dos chats inteligentes está intrinsecamente ligada ao desenvolvimento da Inteligência Artificial (IA) e ao Processamento de Linguagem Natural (PLN). Desde os primeiros experimentos com programas capazes de simular conversas humanas até os avanços mais recentes com modelos como o *Chat GPT*. A evolução desses sistemas reflete o progresso contínuo das tecnologias que sustentam a IA.

O marco inicial na história dos *chatbots* é o ELIZA, desenvolvido por Joseph Weizenbaum no MIT entre 1964 e 1966. ELIZA foi um dos primeiros programas de computador a interagir com usuários em linguagem natural, utilizando um método de substituição de palavras e correspondência de padrões. O *script* mais famoso de ELIZA, chamado "*DOCTOR*", simulava uma conversa com um psicoterapeuta, fazendo com que muitos usuários acreditassem que estavam interagindo com um ser humano. Apesar de suas limitações, ELIZA levantou questões importantes sobre a natureza das interações entre humanos e máquinas e iniciou um novo campo de pesquisa dentro da IA (SILVA, 2024).

Após o desenvolvimento do ELIZA, o próximo grande avanço foi PARRY, criado por Kenneth Colby em 1972. Enquanto ELIZA simulava um psicoterapeuta, PARRY foi projetado para simular o comportamento de uma pessoa com esquizofrenia paranoica. PARRY usava um modelo baseado em regras para simular não apenas respostas linguísticas, mas também uma personalidade e um estado emocional, sendo descrito como "ELIZA com atitude". PARRY foi um passo significativo na tentativa de dar aos chatbots uma personalidade mais complexa e respostas mais dinâmicas (SILVA, 2024).

Com o progresso das tecnologias de IA e o surgimento de dispositivos móveis e assistentes de voz, a década de 2010 marcou o início de uma nova era para os *chatbots*. Empresas como *Apple*, *Amazon* e *Google* lançaram assistentes virtuais capazes de interagir com os usuários em linguagem natural. *Siri* (*Apple*), *Alexa* (*Amazon*) e *Google Assistant* são exemplos de sistemas que utilizam IA para entender comandos de voz e realizar tarefas variadas, desde a busca por informações até o controle de dispositivos domésticos (SILVA, 2024).

Além dos assistentes virtuais, o desenvolvimento de *chatbots* sociais, como o *Xiaoice* da *Microsoft*, trouxe uma nova dimensão para a interação humano-máquina. Chatbots sociais são projetados para manter conversas mais naturais e emocionalmente envolventes, explorando a capacidade de entender e responder de maneira apropriada ao

estado emocional do usuário. Essas inovações evidenciam o progresso significativo em áreas como o Processamento de Linguagem Natural, reconhecimento de voz e aprendizado profundo (SILVA, 2024).

O grande salto tecnológico que transformou a IA conversacional foi a introdução da arquitetura *Transformer*, apresentada no artigo "*Attention is All You Need*" por Vaswani (2017). Diferente das abordagens anteriores que utilizavam convoluções e recorrências, o *Transformer* utiliza o mecanismo de atenção para processar informações de maneira eficiente e em paralelo, o que o torna extremamente eficaz em tarefas de PLN.

Esse avanço abriu o caminho para a criação dos modelos *GPT* (*Generative Pre-trained Transformer*) pela *OpenAI*, que revolucionaram o campo dos chats inteligentes. O *GPT-3*, lançado em 2020, foi um dos primeiros modelos de grande escala a ser amplamente utilizado para diversas aplicações de linguagem natural, como geração de texto, tradução automática e respostas a perguntas. Com 175 bilhões de parâmetros, o *GPT-3* estabeleceu novos padrões de desempenho, sendo capaz de realizar tarefas complexas com alto grau de coerência e contextualidade (OPENAI, 2024).

A evolução continuou com o lançamento do *GPT-3.5* e do *GPT-4*. Este último, introduzido em 2023, trouxe melhorias significativas em termos de compreensão contextual e integração multimodal, permitindo a combinação de texto e imagens em suas respostas. O *GPT-4* representa o auge da tecnologia de IA conversacional até o momento, estabelecendo novos limites para o que é possível em termos de interação homem-máquina (OPENAI, 2024).

A evolução dos chats inteligentes, desde o *ELIZA* até o *Chat GPT*, demonstra um progresso impressionante na capacidade das máquinas de simular a conversa humana. Com o avanço do processamento de linguagem natural, aprendizado profundo e a arquitetura *Transformer*, os chatbots evoluíram de simples sistemas de substituição de texto para modelos altamente sofisticados capazes de interagir de maneira natural e contextual com os usuários. À medida que a tecnologia avança, os chatbots continuarão a desempenhar um papel cada vez mais significativo em nossas vidas, seja como assistentes virtuais, companheiros digitais ou ferramentas de produtividade.

2.2 *Chat GPT*

Chat GPT é um modelo de IA de processamento de linguagem natural (PLN) desenvolvido pela *OpenAI*. Ele utiliza uma arquitetura neural chamada *Transformer* para gerar respostas que se assemelham ao diálogo humano, a partir de grandes quantidades de dados extraídos da internet. Seu treinamento inclui uma técnica chamada *Reinforcement Learning with Human Feedback* (RLHF), que permite à IA aprender padrões e gerar respostas contextualizadas e precisas em diversos cenários (AN; DING; LIN, 2023).

A criação do *Chat GPT* foi liderada pela *OpenAI*, uma organização de pesquisa em inteligência artificial fundada por figuras proeminentes como Elon Musk e Sam Altman. O modelo foi lançado em novembro de 2022 e tem sido amplamente utilizado em aplicações como assistentes virtuais, automação de e-mails e documentos, tradução de idiomas e suporte a sistemas de TI (AN; DING; LIN, 2023).

2.3 Arquitetura Transformer

Os *Transformers*, que alcançaram grande sucesso em diversas áreas da inteligência artificial, como processamento de linguagem natural (PLN), visão computacional e análise de séries temporais, demonstraram seu potencial em modelar dados estruturados em grafos. O artigo se propõe a revisar de forma sistemática as variantes de *Transformers* adaptadas para grafos, discutindo três principais abordagens arquiteturais (MIN et al., 2022).

A arquitetura *Transformer* original, introduzida por Vaswani (2017), utiliza um mecanismo de *self-attention* para modelar relações entre *tokens* de entrada, permitindo aprendizado de dependências de longo alcance. A arquitetura padrão consiste em múltiplas camadas de atenção múltipla (*multi-head attention*) e redes neurais *feed-forward*, aplicadas a cada *token* da sequência. A posição dos *tokens* é codificada por meio de *positional encodings* para capturar a ordem sequencial.

2.3.1 Componentes Básicos da Arquitetura

A arquitetura *Transformer* é composta por dois blocos principais:

Encoder: Responsável por processar a sequência de entrada.

Decoder: Gera a saída, normalmente uma sequência, com base nas representações codificadas da entrada.

Cada um desses blocos é formado por várias camadas idênticas empilhadas, mas com pesos independentes. O *Transformer* típico possui seis camadas no *encoder* e seis no *decoder*, embora esses números possam variar dependendo da implementação Rahali e Akhloufi (2023).

2.3.2 Mecanismo de Atenção

O mecanismo de *self-attention* é o coração do *Transformer*. Ele permite que o modelo se atente a diferentes partes da entrada ao processar uma palavra ou token específico. O processo de *self-attention* funciona calculando através da equação 2.1 um valor de atenção para cada par de *tokens* na sequência de entrada, permitindo que o modelo capture dependências de longo alcance entre *tokens* Rahali e Akhloufi (2023).

$$\text{Attention}(Q, K, V) = F_{\text{distribution}}(F_{\text{alignment}}(K_i, Q)) \cdot V_i \quad (2.1)$$

No contexto do *Transformer*, o estado do decodificador é representado pela *Query* (Q), e os estados ocultos do codificador são representados pelas *Keys* (K) e *Values* (V). O cálculo da atenção envolve a comparação das *queries* com as *keys* para determinar quais valores são mais relevantes para o estado atual.

$F_{\text{alignment}}(K_i, Q)$: Esta é a função de alinhamento, que mede a similaridade ou relevância entre a *query* (Q) e cada *key* (K_i). Isso gera uma pontuação de alinhamento, que reflete quão bem uma *key* específica está alinhada com a *query*.

$F_{\text{distribution}}$: Esta função de distribuição transforma as pontuações de alinhamento em pesos de atenção, que são aplicados aos valores (V). Os métodos mais comuns para essa função de distribuição incluem *SoftMax*, *sigmoid* e funções logísticas.

2.3.2.1 Tipos de Atenção

Atenção Escalonada por Produto Interno (*Scaled Dot-Product Attention*): Essa é a função de atenção mais usada no *Transformer* original, onde os produtos internos entre as representações dos *tokens* são escalonados e normalizados para obter os valores de atenção Rahali e Akhloufi (2023).

Multi-Head Attention: O *Transformer* divide a atenção em várias "cabeças" paralelas, o que permite que o modelo aprenda diferentes representações da sequência em diferentes subespaços Rahali e Akhloufi (2023).

2.3.3 Codificação Posicional (*Positional Encoding*)

Como o *Transformer* processa todos os *tokens* de uma sequência em paralelo, ele não possui uma maneira intrínseca de entender a ordem dos *tokens*. Para lidar com isso, são adicionadas informações de posição à entrada na forma de codificações posicionais. Essas codificações são vetores que representam as posições relativas dos *tokens* na sequência, permitindo ao modelo levar em consideração a ordem Rahali e Akhloufi (2023).

2.3.4 Camadas de *Feed-Forward*

Cada camada de *self-attention* no *Transformer* é seguida por uma camada de rede neural totalmente conectada (ou *feed-forward*) aplicada de maneira independente a cada *token* da sequência. Essas camadas ajudam o modelo a aprender representações mais complexas dos *tokens* Rahali e Akhloufi (2023).

2.3.5 Normalização e Conexões Residuais

Para estabilizar o treinamento, o *Transformer* utiliza a técnica de *layer normalization*, que normaliza as ativações dentro de cada camada. Além disso, há conexões residuais entre as camadas de *self-attention* e *feed-forward*, permitindo que o modelo mantenha informações importantes ao longo das camadas Rahali e Akhloufi (2023).

2.3.6 Capacidades de Pré-Treinamento e Ajuste Fino

O *Transformer* se beneficia muito de grandes volumes de dados durante o pré-treinamento. Uma vez pré-treinado com uma grande quantidade de dados, o modelo pode ser ajustado para tarefas específicas de PLN, como tradução, geração de texto, e outras Rahali e Akhloufi (2023).

3 Metodologia

Neste capítulo, será descrito a metodologia adotada para realização desta pesquisa e para atingir os objetivos estabelecidos neste trabalho. A metodologia englobará as etapas de experimentação, análise e interpretação dos resultados.

3.1 Seleção dos Problemas

A pesquisa se concentra nos problemas de programação propostos pela plataforma *BeeCrowd*, conforme o exemplo demonstrado na Figura 1. Serão utilizadas as mesmas 50 questões selecionadas por Souza (2023). Estas questões foram classificadas anteriormente em três níveis de dificuldade: fácil, intermediário e difícil. Esta classificação e os links diretos para os enunciados dos problemas serão utilizados para garantir a consistência do estudo.

Figura 1 – Questão BeeCrowd

Read four integer values named A, B, C and D. Calculate and print the difference of product A and B by the product of C and D ($A * B - C * D$).

Input
The input file contains 4 integer values.

Output
Print **DIFERENCA** (DIFFERENCE in Portuguese) with all the capital letters, according to the following example, with a blank space before and after the equal signal.

Input Samples	Output Samples
5 6 7 8	DIFERENCA = -26
0 0 7 8	DIFERENCA = -56
5 6 -7 8	DIFERENCA = 86

3.2 Uso do *Chat GPT 4*

Para a resolução dos problemas, será utilizado o modelo *Chat GPT 4*, disponível no ano de 2024. Este modelo é escolhido devido à sua base de treinamento mais ampla e atualizada, potencialmente oferecendo uma capacidade aprimorada de resolução de problemas de programação em comparação com a versão anterior usada por Souza (2023).

3.3 Procedimento de Submissão

Cada problema de programação será submetido ao *Chat GPT 4*, copiando e colando o texto completo do enunciado (descrição, exemplos e restrições), excluindo quaisquer figuras. O *Chat GPT* terá um total de três chances para fornecer uma solução correta para cada problema. As soluções propostas, que serão geradas na linguagem de programação *Python*, serão então submetidas na plataforma *BeeCrowd* para avaliação. Se uma solução for rejeitada a pergunta será submetida novamente, até um total de três tentativas. Problemas não resolvidos após a terceira tentativa serão registrados como tal.

3.4 Limitação de Interações

Devido à limitação de interações do *Chat GPT*, que é de 40 a cada 3 horas, adotaremos uma abordagem de submissão escalonada: 10 perguntas por dia, permitindo um máximo de 3 submissões por pergunta. Esta abordagem totaliza, no máximo, 30 interações diárias com o *Chat GPT*, distribuídas ao longo de 5 dias.

3.5 Documentação e Análise dos Resultados

Cada tentativa de solução será documentada, incluindo o enunciado do problema, o nível de dificuldade, o código de solução proposto pelo *Chat GPT 4*, o status da submissão (aceito ou rejeitado) e o erro específico retornado pela plataforma em caso de rejeição. Esta documentação será organizada em uma planilha para análise comparativa.

3.6 Avaliação e Comparação dos Resultados

Os resultados obtidos com o *Chat GPT 4* serão comparados com os resultados do projeto de Souza (2023), utilizando-se de análise quantitativa e qualitativa. Serão elaborados gráficos para ilustrar a eficácia do *Chat GPT 4* em resolver os problemas em comparação com a versão anterior. A análise buscará identificar melhorias significativas na precisão e eficiência da ferramenta para resolver os problemas propostos pela *BeeCrowd*.

3.7 Hipóteses e Experimentos Complementares

Durante a análise dos resultados preliminares, serão levantadas hipóteses adicionais para explicar o desempenho observado do *Chat GPT 4* em comparação ao *Chat GPT 3.5*. Com o intuito de testar essas hipóteses e obter uma compreensão mais detalhada, serão planejados e executados experimentos complementares.

3.7.1 Análise dos Resultados dos Experimentos Complementares

Os resultados dos experimentos complementares serão analisados e comparados com os resultados obtidos no experimento inicial. A análise incluirá a taxa de acertos, o número de tentativas necessárias para alcançar uma resposta correta e a distribuição dos erros. Esses dados serão armazenados em planilhas e organizados em gráficos, permitindo uma visualização clara das diferenças de desempenho sob as diferentes condições experimentais.

3.8 Resultados Esperados

Esta metodologia visa fornecer um quadro claro e replicável para avaliar a capacidade do *Chat GPT 4* em resolver problemas de programação. A comparação direta com um estudo anterior permitirá avaliar o progresso e as melhorias na tecnologia de IA aplicada à resolução de problemas de programação.

4 Resultados e Discussão

Para alcançar o objetivo da pesquisa, decidiu-se replicar exatamente o experimento proposto por Souza (2023), com a única diferença na versão utilizada, empregando o *Chat GPT 4*.

Durante o experimento, foram utilizadas as mesmas 50 questões propostas pela autora, extraídas da plataforma *BeeCrowd*, sem modificar seus enunciados e sem propor a análise de figuras demonstrativas, caso existissem.

Vale destacar que a plataforma *BeeCrowd* classifica seus problemas por nível de dificuldade, de 1 a 10. Assim, foram selecionadas 5 questões de cada nível, totalizando 50 problemas. Para facilitar a comparação com os problemas do *LeetCode*, a autora redistribuiu as questões em três níveis semânticos:

- Fácil: questões do nível 1 ao 4;
- Intermediário: questões do nível 5 ao 7;
- Difícil: questões do nível 8 ao 10.

Neste projeto, manteve-se essa classificação para facilitar a comparação dos resultados, apresentados em gráficos.

É importante salientar que na plataforma *BeeCrowd*, o nível de dificuldade é estimado usando uma variação do *ELO Rating System*, explicado por Glickman e Jones (1999). Esse sistema utiliza o número de vezes que o problema foi resolvido para determinar sua dificuldade. Quanto mais esses problemas forem resolvidos com poucas tentativas, eles ganham uma pontuação mais baixa, sendo considerados mais fáceis.

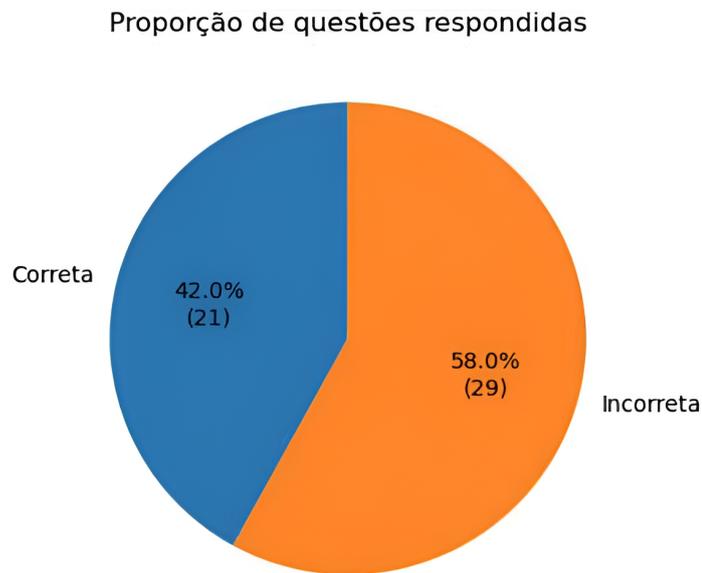
4.1 *Chat GPT 4* em comparação ao *Chat GPT 3.5*

Nos testes realizados por Souza (2023) foram obtidos os seguintes resultados. Vinte e uma questões corretas das cinquenta questões sendo consideradas:

- 16 fáceis;
- 3 intermediárias;
- 2 difíceis.

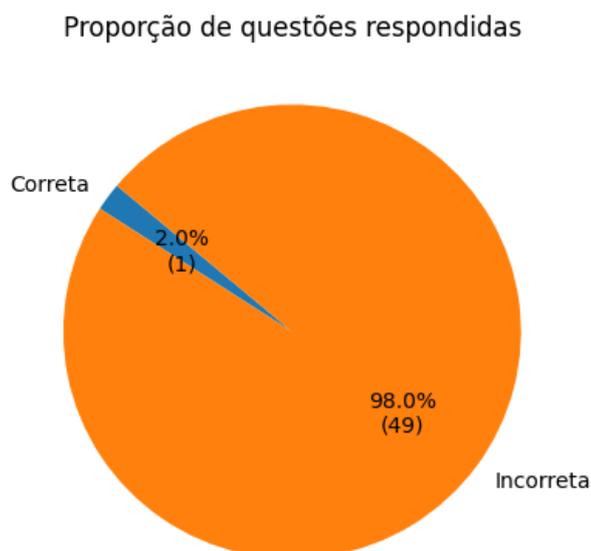
Na Figura 2, observa-se a proporção de acertos representada em um gráfico de pizza, indicando que 42% das questões foram respondidas corretamente.

Figura 2 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com *Chat GPT 3.5*.



Fonte: Souza (2023)

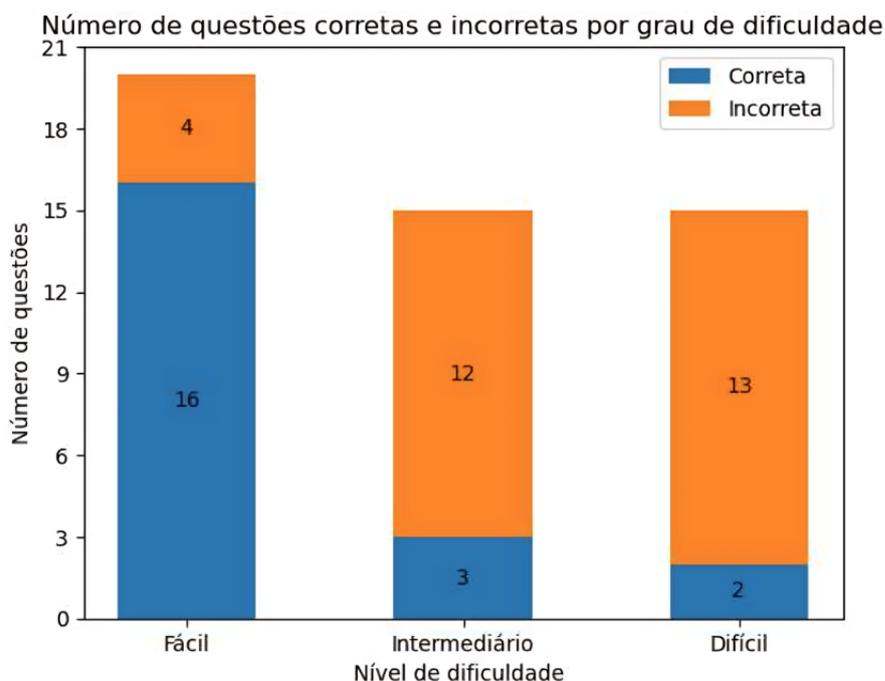
Figura 3 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com *Chat GPT 4*.



Com esses resultados em mente, foi realizado o mesmo experimento com o *Chat GPT 4*, obtendo-se como resultado apenas uma questão correta das 50 propostas.

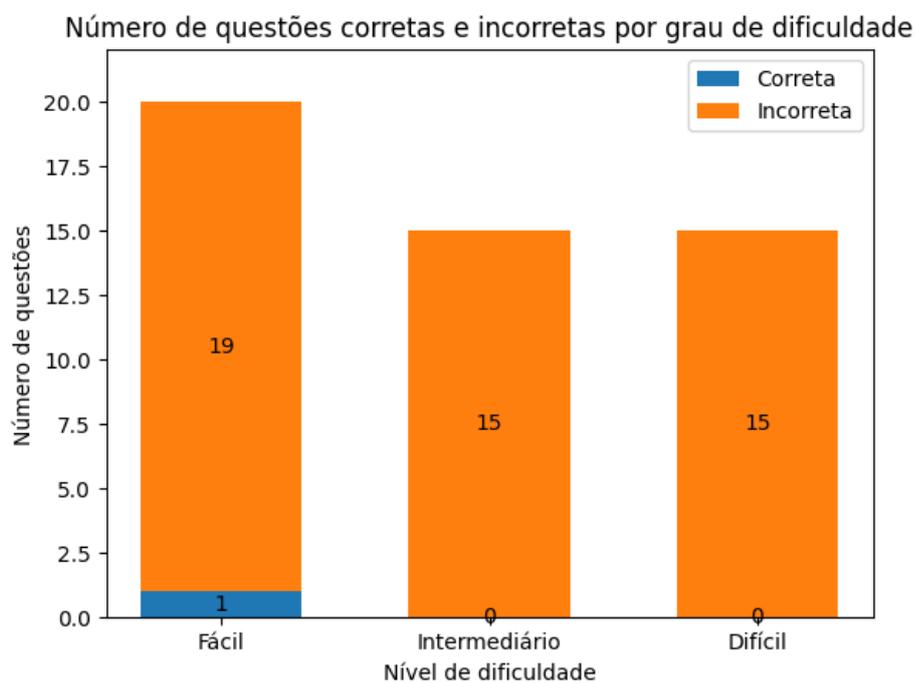
Durante a realização do experimento, foi gerado o gráfico apresentado na Figura 3 para fins de comparação.

Figura 4 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com *Chat GPT 3.5*.



Fonte: Souza (2023)

Figura 5 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4*.



Conforme observado, obtivemos uma taxa de acerto significativamente menor em comparação com o experimento original. Apenas 2% dos problemas propostos foram considerados corretos pela plataforma, em contraste com os 42% obtidos no experimento com o Chat GPT 3.5 realizado por Souza (2023).

Além disso, podemos comparar as taxas de acertos e erros por grau de dificuldade. Na Figura 4, a proporção de acertos é bem maior no nível fácil, com 16 questões corretas das 20 propostas. Observamos também que 3 das 15 questões intermediárias e 2 das 15 questões difíceis foram respondidas corretamente.

Em contrapartida, na Figura 5, observa-se que apenas uma questão do nível fácil foi aprovada pela plataforma entre as 50 questões propostas.

Além disso, é possível comparar a taxa de acertos por número de tentativas e grau de dificuldade das questões. No experimento original, foram necessárias duas tentativas para gerar o código correto nas questões que foram respondidas corretamente. A proporção de acertos por tentativas e nível de dificuldade pode ser observada na Figura 6.

Em comparação apresenta-se o mesmo gráfico para o experimento realizado com o *Chat GPT 4*, exposto na Figura 7.

Observa-se que, além de apenas uma das respostas do *Chat GPT 4* ter obtido êxito ao ser submetida à plataforma, essa resposta foi bem-sucedida apenas na segunda tentativa de submissão da questão ao modelo.

Adicionalmente, destaca-se que a questão bem-sucedida era de nível fácil. Enquanto o *Chat GPT 3.5* por sua vez, obteve êxito em algumas questões de nível intermediário (3 respostas) e difícil (2 respostas), além das 16 respostas corretas no nível fácil, sendo que 14 delas foram submetidas ao modelo apenas uma vez.

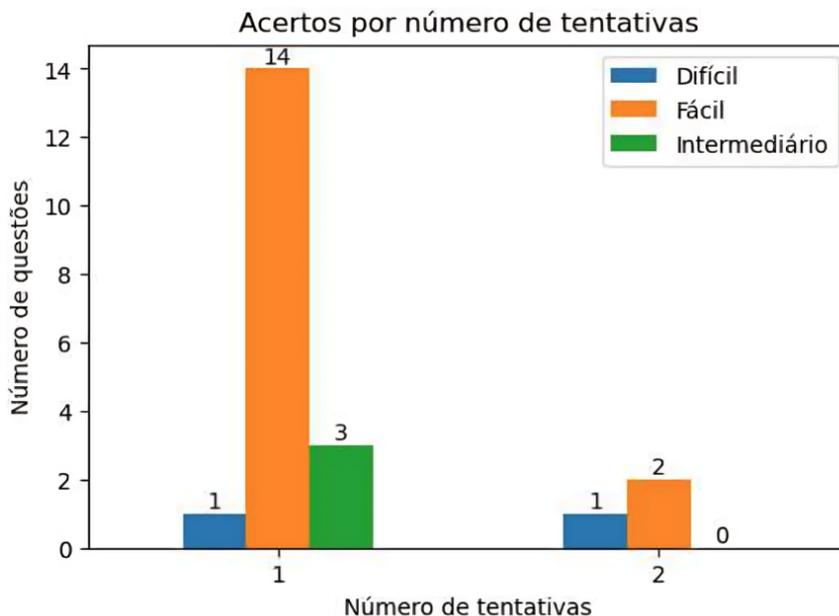
Por fim, é possível observar a proporção dos erros ao submeter as respostas. Para facilitar a comparação, foram mantidos os mesmos padrões de cores para erros semelhantes, além de novas cores para evidenciar erros distintos.

Durante a comparação, constatou-se que o *Chat GPT 3.5* obteve apenas 11 questões com a resposta completamente errada (*Wrong answer (100%)*), conforme a Figura 8, em comparação com as 35 questões com respostas completamente erradas do *Chat GPT 4*, apresentadas na Figura 9.

Quanto aos erros de tempo de execução (*Runtime error*) o número foi semelhante, com 6 respostas erradas para o *Chat GPT 3.5*, apresentados na Figura 8, e 7 respostas para o *Chat GPT 4*, com a proporção apresentada na Figura 9.

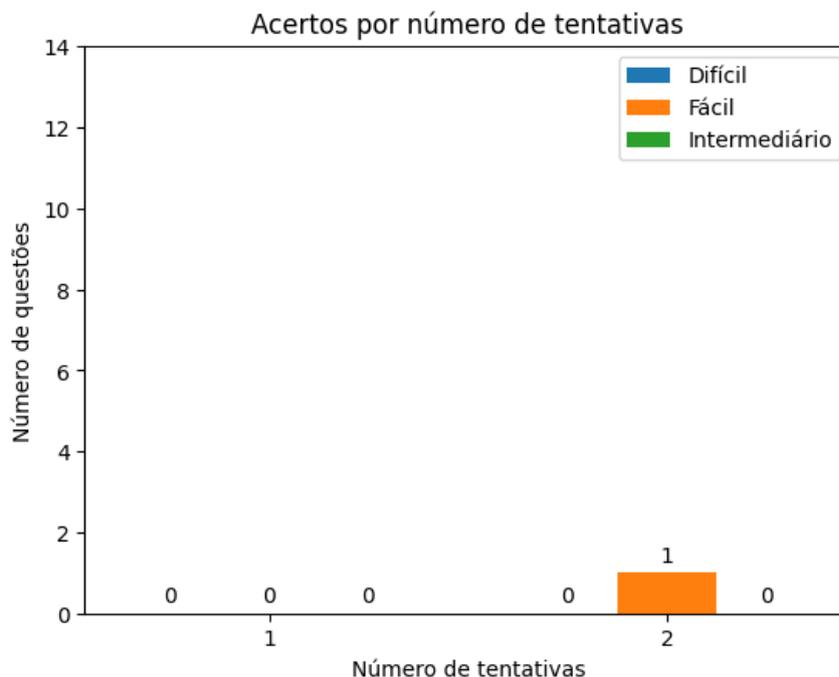
Além disso, foram observados outros erros que não se repetiram nas respostas dos dois modelos. No *Chat GPT 3.5* tivemos 1 erro de estouro de memória (*Memory limit exceeded*) e 11 questões que ultrapassaram o limite de tempo (*Time limit exceeded*), as

Figura 6 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com *Chat GPT 3.5*.



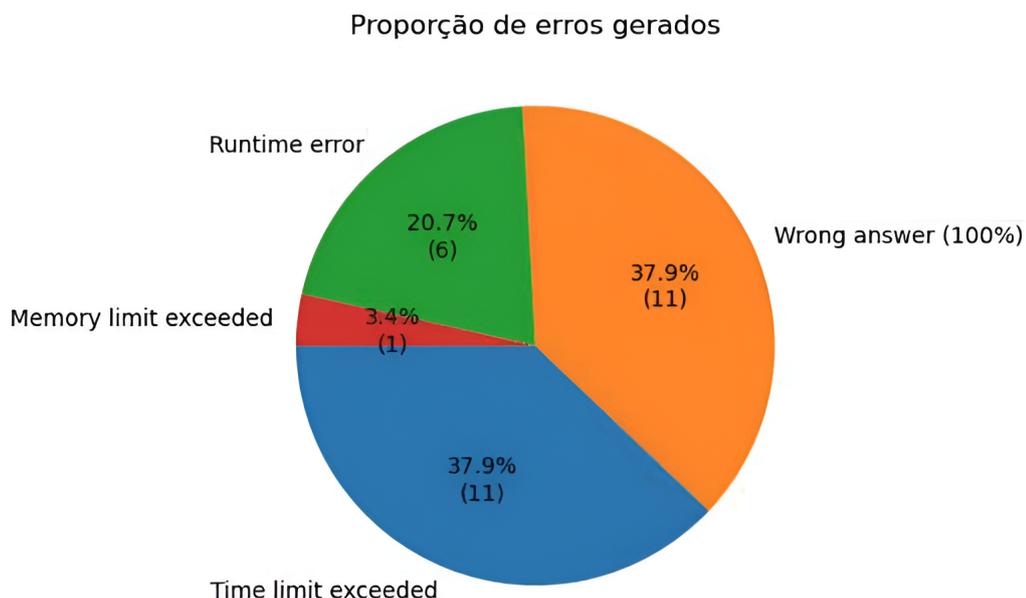
Fonte: Souza (2023)

Figura 7 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4*.



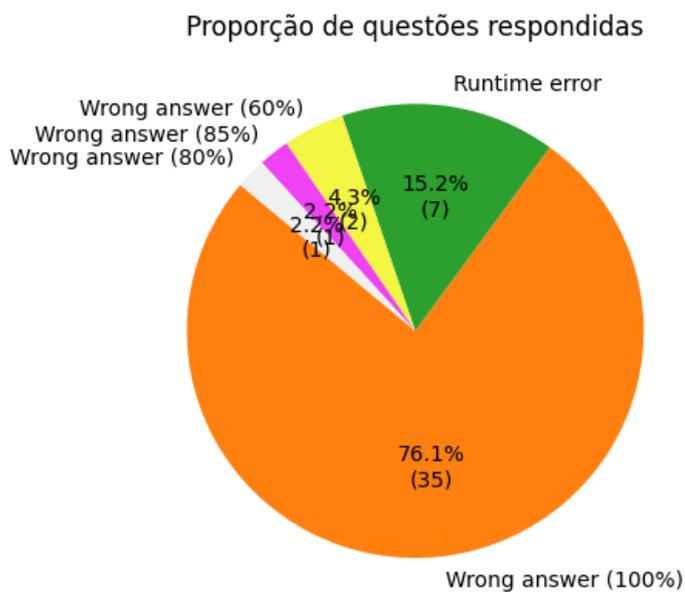
proporções são encontradas no gráfico da Figura 8. Já no *Chat GPT 4* foram apresentados alguns erros na resposta apresentando uma porcentagem de quão diferente está da resposta esperada pelo teste realizado na plataforma. O *Chat GPT 4* apresenta respostas erradas (*Wrong answer*) também nas proporções de 80%, 85% e 60%, conforme a Figura 9.

Figura 8 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com *Chat GPT 3.5*.



Fonte: Souza (2023)

Figura 9 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com *Chat GPT 4*.



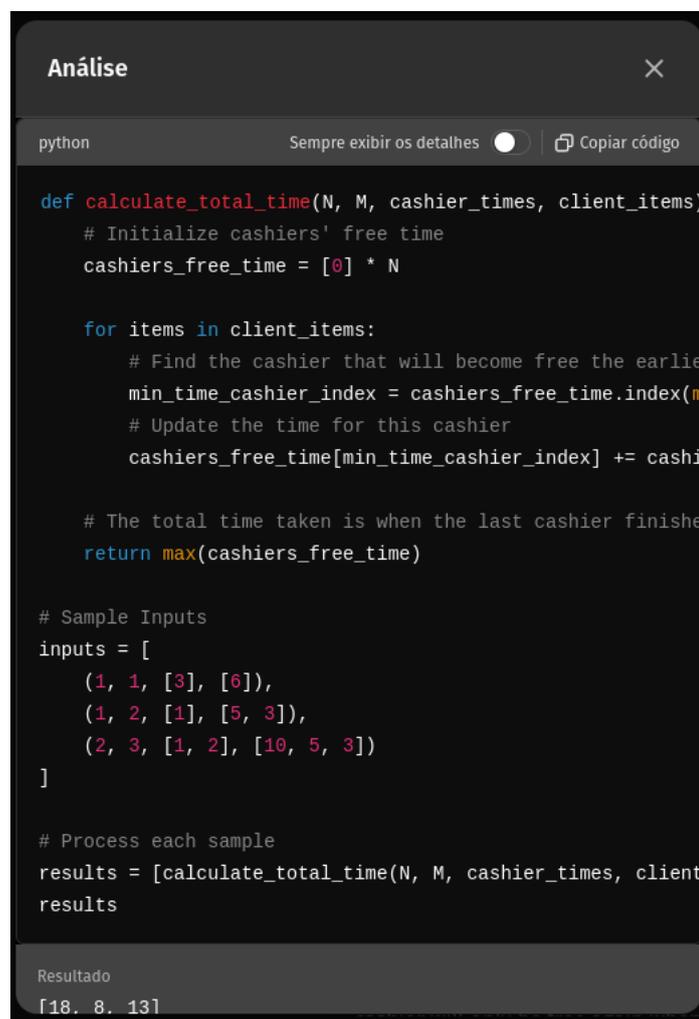
4.2 Hipótese levantada para o baixo desempenho do *Chat GPT 4*

No decorrer do estudo, foi levantada uma hipótese para explicar os resultados insatisfatórios do modelo *Chat GPT 4* em comparação ao seu predecessor.

Observou-se que, embora o *Chat GPT 4* gerasse respostas erradas para as mesmas perguntas propostas ao *Chat GPT 3.5*, suas respostas eram mais completas, incluindo autoanálises e inferências sobre seus próprios erros durante a geração da resposta.

Verificou-se que o código gerado pelo *Chat GPT 4* frequentemente aplicava o exemplo demonstrativo apresentado na questão a uma variável fixa, em vez de tratar o exemplo como um valor a ser recebido pelo usuário através de uma entrada (*input*) no *console*. Essa característica pode ser exemplificada pelo código gerado pelo modelo, conforme ilustrado na Figura 10, para a questão descrita no Anexo A.

Figura 10 – Exemplo de código gerado com *Chat GPT 4*.



```
python Sempre exibir os detalhes  Copiar código

def calculate_total_time(N, M, cashier_times, client_items)
# Initialize cashiers' free time
cashiers_free_time = [0] * N

for items in client_items:
# Find the cashier that will become free the earliest
min_time_cashier_index = cashiers_free_time.index(min(cashiers_free_time))
# Update the time for this cashier
cashiers_free_time[min_time_cashier_index] += cashier_times[items[0]-1][1]

# The total time taken is when the last cashier finishes
return max(cashiers_free_time)

# Sample Inputs
inputs = [
(1, 1, [3], [6]),
(1, 2, [1], [5, 3]),
(2, 3, [1, 2], [10, 5, 3])
]

# Process each sample
results = [calculate_total_time(N, M, cashier_times, client_items) for client_items in inputs]
results
```

Resultado
[18. 8. 13]

O código gerado pelo *Chat GPT 4* está completamente descrito no Anexo B.

Com base nisso, foram propostos dois experimentos complementares, seguindo o mesmo roteiro da metodologia original, mas usando apenas 5 questões aleatórias das 50 utilizadas no experimento inicial. A diferença está nas seguintes condições: no primeiro experimento, as 5 questões não deveriam conter os exemplos propostos, apenas o enunciado. No segundo experimento, seria acrescentado um texto indicando que os exemplos a seguir deveriam ser inseridos no *console*.

Para o segundo caso foi adicionado a seguinte frase "A seguir temos exemplos de como deverão ser as entradas e saídas no console". Para os problemas descritos em inglês traduziu-se a frase para "*Below we have examples of what the inputs and outputs on the console should look like*".

Conforme o experimento obteve-se gráficos para ambos os casos.

Figura 11 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com *Chat GPT 4* sem indicar o exemplo.

Proporção de questões respondidas

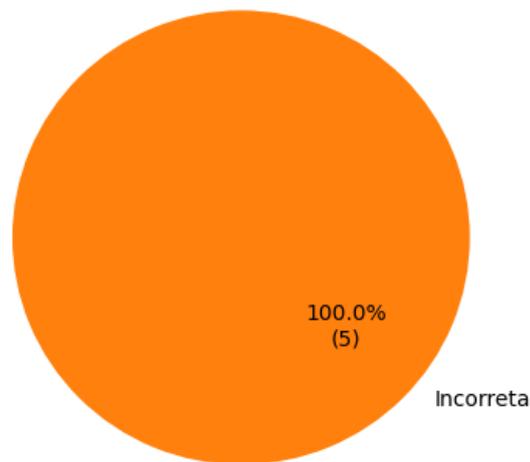
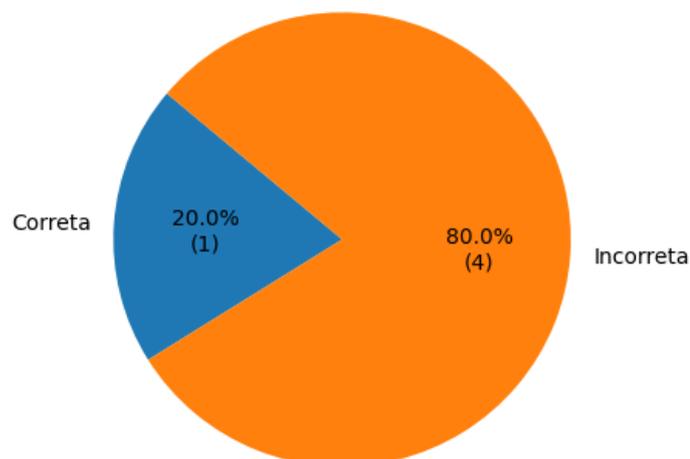


Figura 12 – Gráfico com proporção de problemas respondidos na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.

Proporção de questões respondidas



Primeiramente, foi obtida a taxa de acerto. No experimento sem a inclusão de exemplos, não se obteve nenhuma resposta correta, conforme ilustrado na Figura 11.

No entanto, no experimento que incluiu exemplos, observou-se que houve um acerto entre as questões propostas, conforme ilustrado na Figura 12.

Figura 13 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4* sem indicar o exemplo.

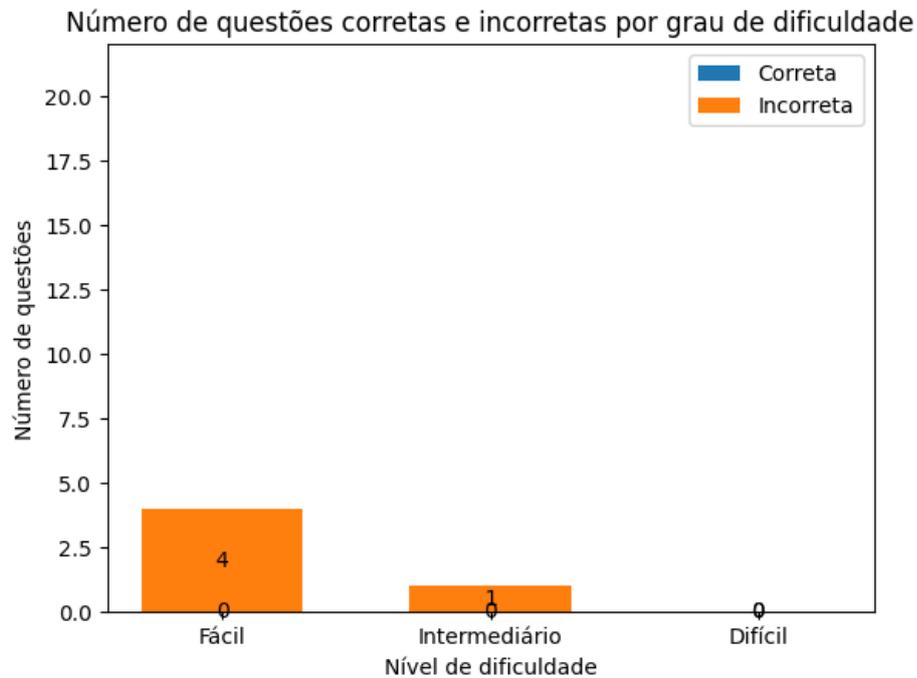
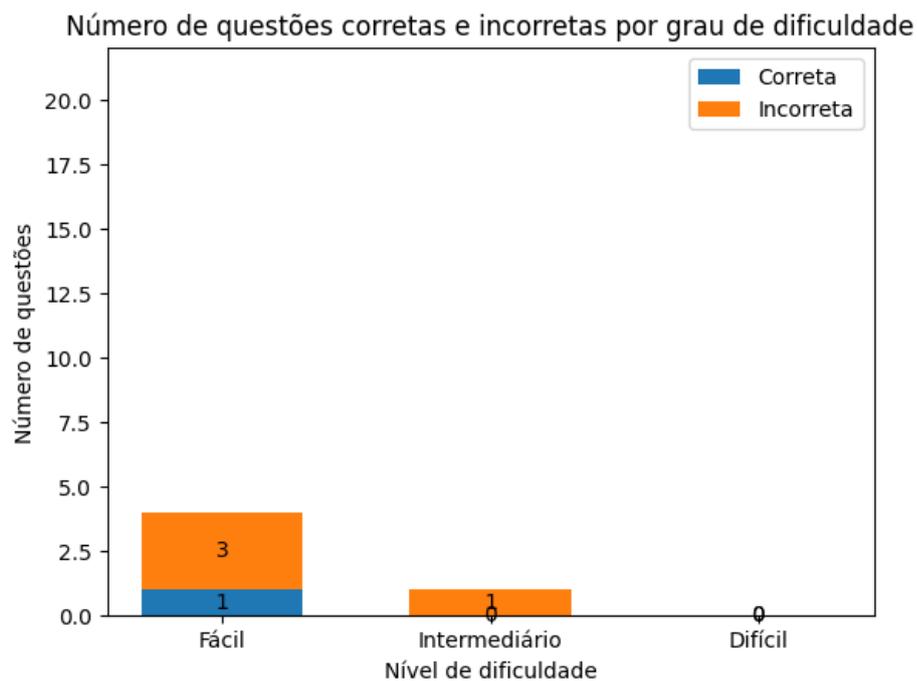


Figura 14 – Gráfico com a proporção de acertos e erros por grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.



Também é possível observar a proporção de acertos e erros por nível de dificuldade. Para o teste sem os exemplos, o gráfico apresentado na Figura 13 demonstra que não houve nenhum acerto.

Figura 15 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4* sem indicar o exemplo.

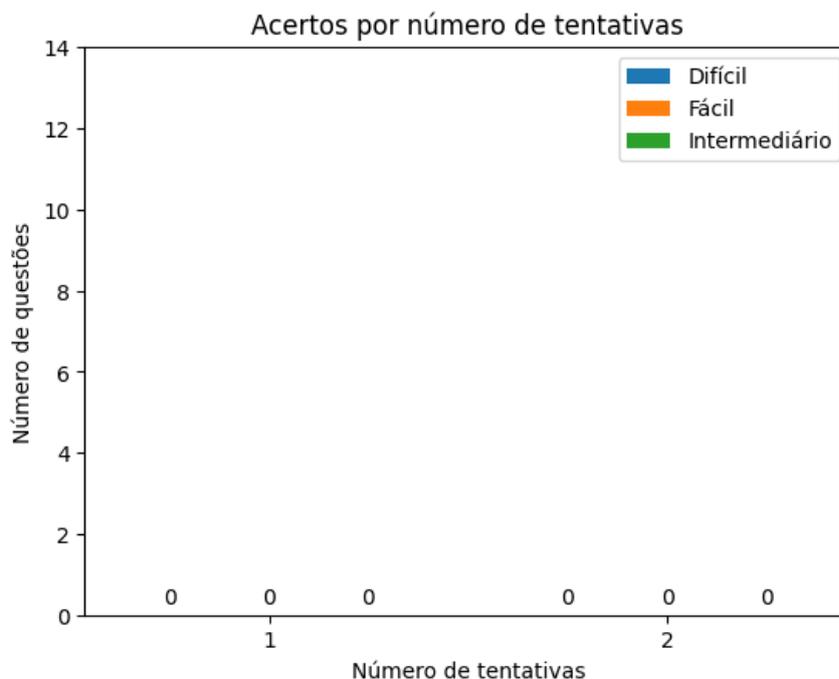
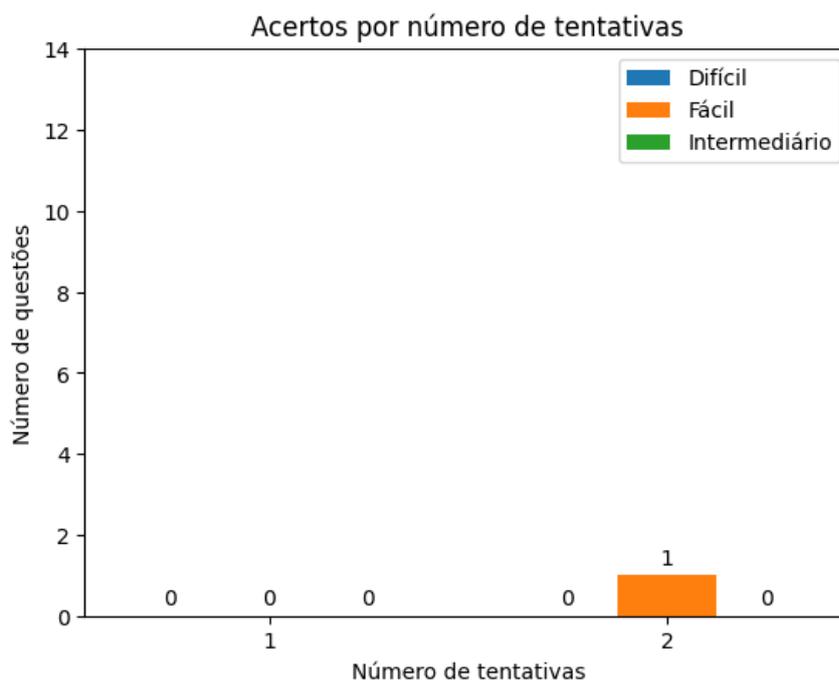


Figura 16 – Gráfico com número de acertos por número de tentativas e grau de dificuldades na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.



Em contrapartida, o gráfico que mostra a proporção de acertos e erros por nível de dificuldade para os casos em que os exemplos foram incluídos, apresentado na Figura 14, revela que a questão acertada era de nível fácil.

Figura 17 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com *Chat GPT 4* sem indicar o exemplo.

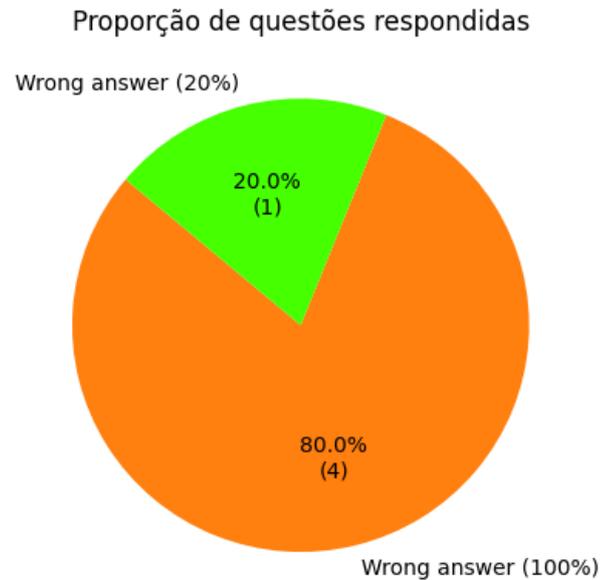
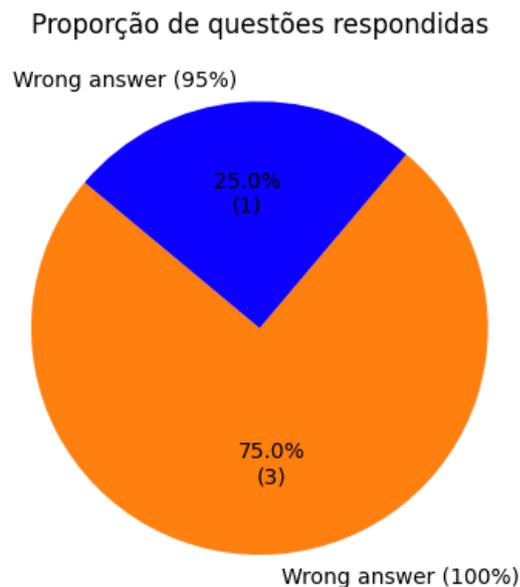


Figura 18 – Gráfico com os erros gerados pelas respostas submetidas à plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.



Também foi analisado o número de tentativas necessárias para as questões corretas em cada experimento. Na Figura 15 o, o gráfico está vazio, uma vez que não houve nenhum acerto no experimento sem exemplos. Por outro lado, no experimento em que os exemplos

foram incluídos, conforme apresentado na Figura 16, observa-se que a questão correta, além de ser de nível fácil, foi validada pelo *BeeCrowd* apenas na segunda tentativa.

Por fim é possível evidenciar os erros gerados por ambos os experimentos propostos, tanto na Figura 17 quanto na Figura 18. Na Figura 18 foram encontrados, dentre o total de erros, 75% sendo "Wrong answer 100%" e 25% sendo "Wrong answer 95%", e na Figura 17 apresentou-se 80% de erro "Wrong answer 100%" e 20% de "Wrong answer 20%".

Finalmente, após a análise dos resultados, decidiu-se aprofundar a exploração do caso que inclui exemplos, uma vez que este apresentou um resultado correto ao ser submetido à plataforma.

4.3 Comparação entre *Chat GPT 3.5* e *Chat GPT 4* com o texto indicando os exemplos

Com o objetivo de aprofundar a exploração do experimento número dois, descrito na seção 4.2, decidiu-se realizar o experimento novamente para todas as 50 questões, incluindo o texto que indica os exemplos de entrada e saída a serem inseridos no *console*.

Com isso, foram gerados os mesmos gráficos obtidos anteriormente, com o objetivo de comparar os resultados do *Chat GPT 3.5* e do *Chat GPT 4*, conforme demonstrado na seção 4.1

O gráfico gerado para a proporção de acertos está apresentado na Figura 19. É possível perceber um número significativamente maior de acertos em comparação ao gráfico da Figura 3, que representa a proporção de acertos obtida pelo *Chat GPT 4*. No entanto, o número de acertos ainda é menor em comparação à quantidade de questões respondidas corretamente pelo *Chat GPT 3.5*, conforme indicado na Figura 2.

Além disso, foi criado o gráfico apresentado na Figura 20, que representa a quantidade de acertos e erros proporcionalmente divididos entre os três níveis de dificuldade.

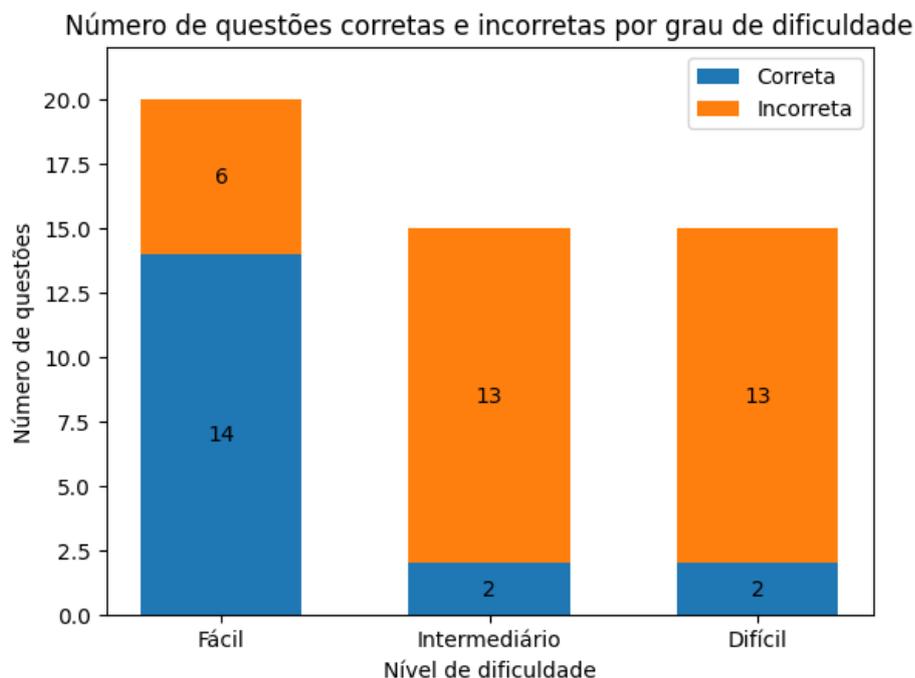
O gráfico permite dimensionar o número de acertos para cada nível de dificuldade. Em comparação com o primeiro experimento, apresentado no gráfico da Figura 5, observa-se que 14 questões foram respondidas corretamente no nível fácil. Adicionalmente, foram respondidas corretamente duas questões no nível intermediário e duas no nível difícil.

Em relação às respostas obtidas no experimento original, apresentadas na Figura 4, o desempenho do *Chat GPT 4* continuou inferior, embora próximo aos resultados do *Chat GPT 3.5*. O *Chat GPT 4* obteve 14 respostas corretas no nível fácil, em comparação com 16 respostas corretas pelo *Chat GPT 3.5*. No nível intermediário, o *Chat GPT 4* acertou 2 questões, enquanto o modelo anterior acertou 3. No nível difícil, ambos os modelos empataram com duas questões respondidas corretamente.

Figura 19 – Gráfico com proporção dos 50 problemas respondidos na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.



Figura 20 – Gráfico com a proporção de acertos e erros por grau de dificuldades dos 50 problemas respondidos na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.

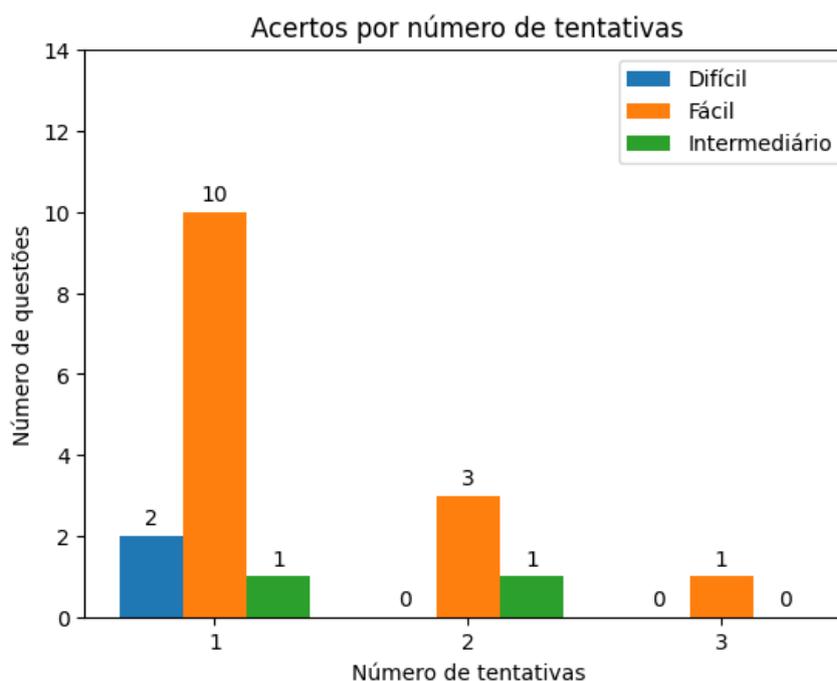


A taxa de acertos para cada nível de dificuldade, por número de tentativas, é apresentada na Figura 21. Para o nível fácil, houve 10 acertos na primeira tentativa, 3 acertos na segunda tentativa e 1 acerto na terceira tentativa.

Esses resultados são semelhantes aos do experimento original, que também apresentou a maior taxa de acertos na primeira tentativa. Comparando com a Figura 6, observa-se que todos os acertos foram obtidos em apenas duas tentativas: 14 na primeira e 2 na segunda para o nível fácil. Em comparação com a Figura 7, percebe-se que os acertos ocorreram na primeira tentativa de submissão, em oposição à única resposta correta obtida na segunda submissão ao *Chat GPT 4*.

Para o nível intermediário, houve 1 acerto na primeira tentativa e 1 acerto na segunda tentativa. No nível difícil, ambos os acertos ocorreram na primeira tentativa. Este resultado inverteu o padrão observado no primeiro experimento, onde as 3 questões intermediárias corretas foram acertadas na primeira submissão ao *Chat GPT 3.5*, e as questões de nível difícil foram acertadas uma na primeira submissão e a outra na segunda.

Figura 21 – Gráfico com número de acertos por número de tentativas e grau de dificuldades dos 50 problemas respondidos na plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.

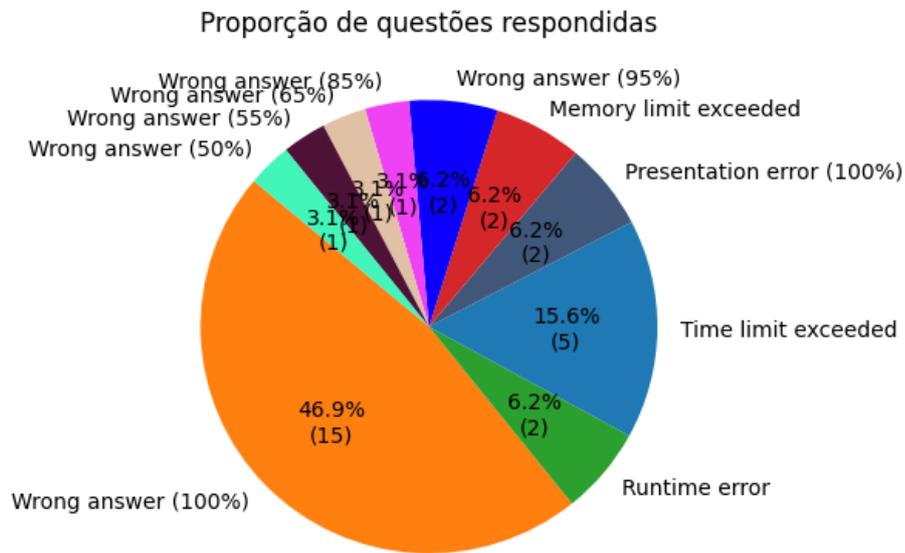


Por último, é possível observar a distribuição dos erros obtidos na plataforma *BeeCrowd*. A Figura 22 apresenta a proporção dos erros.

Em comparação com a Figura 8, observa-se uma taxa menor de erros por ultrapassar o limite de tempo, sendo apenas 5 erros em comparação aos 11 erros obtidos pelo *Chat GPT 3.5*.

Quanto aos erros de estouro de memória, houve 2 erros, em oposição ao único erro desse tipo apresentado pelo modelo anterior. Nos erros de tempo de execução, o *Chat GPT 4* apresentou 2 erros, comparados aos 6 erros do modelo anterior.

Figura 22 – Gráfico com os erros gerados pelas respostas dos 50 problemas respondidos submetidas à plataforma BeeCrowd com *Chat GPT 4* indicando o exemplo.



Adicionalmente, o *Chat GPT 4* teve 15 respostas completamente erradas (100% erradas), 2 respostas com 95% de erro, e 4 questões com erros nas proporções de 50%, 55%, 65%, e 85%. Também foram observados erros de apresentação (*Presentation error* 100%), que não são necessariamente erros no código, mas indicam que a apresentação da resposta está diferente do esperado pelo problema proposto na plataforma.

5 Conclusão

O presente estudo teve como objetivo comparar o desempenho do *Chat GPT 3.5* e do *Chat GPT 4* na resolução de problemas de programação propostos pela plataforma *BeeCrowd*. A metodologia adotada foi replicar o experimento original realizado por Souza (2023) com o *Chat GPT 3.5*, utilizando as mesmas 50 questões, classificadas em três níveis de dificuldade: fácil, intermediário e difícil.

Os resultados obtidos indicaram que o *Chat GPT 4* apresentou um desempenho inferior ao *Chat GPT 3.5*. No experimento original, o *Chat GPT 3.5* obteve 21 respostas corretas, sendo 16 questões de nível fácil, 3 intermediárias e 2 difíceis, representando uma taxa de acerto de 42%. Em contrapartida, o *Chat GPT 4* conseguiu resolver corretamente apenas uma questão das 50 propostas, resultando em uma taxa de acerto de 2%.

Diante desse desempenho, foram realizados experimentos complementares para explorar possíveis hipóteses para o baixo desempenho do *Chat GPT 4*. Observou-se que o modelo 4 frequentemente aplicava exemplos demonstrativos a variáveis fixas em vez de tratá-los como valores de entrada fornecidos pelo usuário. Esse comportamento foi investigado em dois experimentos adicionais: um sem a inclusão de exemplos nos problemas e outro com a inclusão de instruções explícitas sobre os exemplos.

Os experimentos adicionais mostraram que, ao incluir instruções explícitas sobre os exemplos, o *Chat GPT 4* teve um desempenho melhor, respondendo corretamente a 14 questões de nível fácil, 2 intermediárias e 2 difíceis. Contudo, mesmo com essa melhoria, o desempenho geral do *Chat GPT 4* ainda foi inferior ao do *Chat GPT 3.5*.

A análise dos erros gerados indicou que o *Chat GPT 4* apresentou uma maior quantidade de respostas completamente erradas (35 questões) e erros de tempo de execução (7 questões) em comparação ao *Chat GPT 3.5*, que teve 11 respostas completamente erradas e 6 erros de tempo de execução. Além disso, o *Chat GPT 4* mostrou erros específicos de porcentagem de erro na resposta (80%, 85% e 60%), não observados no *Chat GPT 3.5*.

Em resumo, apesar das melhorias observadas com a inclusão de instruções explícitas sobre exemplos, o *Chat GPT 4* não superou o desempenho do *Chat GPT 3.5* na resolução de problemas de programação da plataforma *BeeCrowd*. Estes resultados sugerem que, embora o *Chat GPT 4* tenha uma capacidade de geração de respostas mais completa e detalhada, a sua eficácia na resolução de problemas de programação pode ser prejudicada por abordagens inadequadas de processamento de exemplos demonstrativos. Esses achados ressaltam a importância de futuras investigações para aprimorar a aplicação do *Chat GPT* em contextos de resolução de problemas de programação.

5.1 Trabalhos Futuros

Para ampliar os resultados deste estudo, sugere-se realizar experimentos futuros utilizando outras ferramentas de IA generativas, como o *Gemini* do *Google* e o *Microsoft Copilot*. Esses experimentos seguirão a mesma metodologia aplicada aqui, utilizando os mesmos 50 problemas da *BeeCrowd* para garantir a comparabilidade dos resultados.

Como trabalho adicional também sugere-se repetir o estudo para versões mais atualizadas do *Chat GPT*, com a finalidade de continuar aferindo sua evolução com o passar do tempo.

Esses estudos futuros poderão fornecer informações valiosas sobre as capacidades e limitações das diferentes ferramentas de IA generativas, contribuindo para o avanço contínuo das tecnologias de IA na programação.

Referências

- AN, J.; DING, W.; LIN, C. Chatgpt: tackle the growing carbon footprint of generative ai. *Nature*, Nature, v. 615, n. 7953, p. 586–586, 2023.
- GLICKMAN, M. E.; JONES, A. C. Rating the chess rating system. *CHANCE-BERLIN THEN NEW YORK- 12*, 1999.
- HAI, H. N. Chatgpt: The evolution of natural language processing. *Authorea Preprints*, Authorea, 2023.
- MIN, E.; CHEN, R.; BIAN, Y.; XU, T.; ZHAO, K.; HUANG, W.; ZHAO, P.; HUANG, J.; ANANIADOU, S.; RONG, Y. Transformer for graphs: An overview from architecture perspective. *arXiv preprint arXiv:2202.08455*, 2022.
- MOTTA, I. dos S.; ALVES, J. R.; ANDRADE, T. L. A utilização do chat gpt para desenvolvimento das aplicações web. *Seminário de Pesquisa e Iniciação Científica; Centro Universitário de Barra Mansa; <http://aete.ubm.br:8081/repositorio/handle/123456789/361>*, 2023.
- OPENAI. *Models*. 2024. Acessado: 22/08/2024. Disponível em: <<https://platform.openai.com/docs/models>>
- RAHALI, A.; AKHLOUFI, M. A. End-to-end transformer-based models in textual-based nlp. *AI*, MDPI, v. 4, n. 1, p. 54–110, 2023.
- SEIKKINEN, E. How to use chatgpt for programming. *Metropolia University of Applied Sciences, Bachelor of Engineering, Information and Communications Technology. Finlândia*, 2023.
- SILVA, C. A. G. da; RAMOS, F. N.; MORAES, R. V. de; SANTOS, E. L. dos. Chatgpt: Challenges and benefits in software programming for higher education. *Sustainability* 2024, 16(3), 1245; <https://doi.org/10.3390/su16031245>, 2024.
- SILVA, H. P. R.; SANTOS, C. H. M. da S.; SOUZA, I. R. de; CALDAS, A. G. S.; SANTOS, I. G. de S. Desenvolvimento de jogos 2d de plataforma: Explorando unity e chat gpt para criação de códigos dinâmicos. *Anais do XXX Congresso Brasileiro de Informática na Educação*, 2023.
- SILVA, M. A. B. da. Do eliza ao chatgpt: História e evolução da inteligência artificial. *Trabalho de Conclusão de Curso; Pontifícia Universidade Católica de Goiás*, 2024.
- SOARES, M. Impacto do chat gpt na sociedade. *The Trends Hub*, n. 3, 2023.
- SOUZA, D. L. de L. Estudo de caso: Uso do chatgpt para resolução de problemas de programação. 2023. *Trabalho de Conclusão Curso - Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática - Universidade Federal de Campina Grande. Campina Grande, PB. <http://dspace.sti.ufcg.edu.br:8080/xmlui/handle/riufcg/30461>*, 2023.

SURAMEERY, N. M. S.; SHAKOR, M. Y. Use chat gpt to solve programming bugs. *International Journal of Information Technology and Computer Engineering*; 10.55529/ijitc.31.17.22, 2023.

VASWANI, A. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Anexos

ANEXO A – Exemplo de questão proposta ao *Chat GPT4*

Today is the inauguration of a huge supermarket in your town, and everyone are excited about the low prices promised.

This supermarket has N cashiers, identified by numbers from 1 to N , where each cashier takes a specific amount of time v_i to process an item from a client. Therefore, if a client has c_j items on his basket, a specific cashier will take $v_i * c_j$ seconds to process all of the items from this client.

When a client enters the line to be attended he waits until a cashier is free. If more than one cashier are free at the same time, he will be attended by the cashier with the lowest identification number. This cashier will only be free again when he finishes processing all of the clients items.

There are M clients on the line to be attended, each with a specific number of items on his basket. Given the information about the cashiers and the clients, the manager asked your help to find out how long it will take so all the clients are attended.

Input The first line of input has two integers N and M , indicating the number of cashiers and clients, respectively ($1 \leq N \leq M \leq 104$).

Following there will be N integers v_i , indicating how long the i -th cashier takes to process an item ($1 \leq v_i \leq 100$, for every $1 \leq i \leq N$).

Following there will be M integers c_j , indicating how many items the j -th client has ($1 \leq c_j \leq 100$, for every $1 \leq j \leq M$).

Output Print a line containing an integer, indicating how long it will take so all the clients are attended.

Input Samples

1 1

3

6

Output Samples

18

Input Samples

1 2

1

5 3

Output Samples

8

Input Samples

2 3

1 2

10 5 3

Output Samples

13

ANEXO B – Exemplo de código resposta do *Chat GPT4*

```
1 def calculate_total_time(N, M, cashier_times, client_items):
2     # Initialize cashiers' free time
3     cashiers_free_time = [0] * N
4
5     for items in client_items:
6         # Find the cashier that will become free the earliest
7         min_time_cashier_index = cashiers_free_time.index(min(cashiers_free_time))
8         # Update the time for this cashier
9         cashiers_free_time[min_time_cashier_index] += cashier_times[min_time_cashier_index] * items
10
11    # The total time taken is when the last cashier finishes processing
12    return max(cashiers_free_time)
13
14 # Sample Inputs
15 inputs = [
16     (1, 1, [3], [6]),
17     (1, 2, [1], [5, 3]),
18     (2, 3, [1, 2], [10, 5, 3])
19 ]
20
21 # Process each sample
22 results = [calculate_total_time(N, M, cashier_times, client_items) for N, M, cashier_times, client_items in inputs]
23 results
24
```