UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

DANIEL CARLOS ALVES DE MELO

ANALISANDO LLMS NA RESOLUÇÃO DE PROBLEMAS DE COMPETIÇÕES DE PROGRAMAÇÃO: UM ESTUDO COM CHATGPT E BARD

CAMPINA GRANDE - PB

2023

DANIEL CARLOS ALVES DE MELO

ANALISANDO LLMS NA RESOLUÇÃO DE PROBLEMAS DE COMPETIÇÕES DE PROGRAMAÇÃO: UM ESTUDO COM CHATGPT E BARD

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

Orientador : Rohit Gheyi

CAMPINA GRANDE - PB

2023

DANIEL CARLOS ALVES DE MELO

ANALISANDO LLMS NA RESOLUÇÃO DE PROBLEMAS DE COMPETIÇÕES DE PROGRAMAÇÃO: UM ESTUDO COM CHATGPT E BARD

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Rohit Gheyi

Orientador – UASC/CEEI/UFCG

Eanes Torres

Examinador – UASC/CEEI/UFCG

Melina Mongiovi

Professora da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 16 de NOVEMBRO de 2023.

CAMPINA GRANDE - PB

RESUMO

Com a recente adoção de Inteligências Artificiais Generativas no campo da Ciência da Computação, surge o questionamento dos limites dessas ferramentas na geração de código e no desempenho desse código na resolução de problemas de programação de alta complexidade, como desafios envolvendo programação dinâmica. Neste estudo, analisaremos o desempenho de dois modelos de IA(Inteligência Artificial), o Bard e o ChatGPT, ao submetê-los a 83 problemas de programação de diversos níveis de complexidade. Utilizaremos os prompts de ambas as ferramentas em um estudo de caso comparativo que aplicará o código gerado por esses modelos em problemas de programação obtidos de juízes online, incluindo Codeforces, Atcoder, CodeChef e LeetCode. Compararemos os resultados desses modelos em competições online realizadas regularmente pelas plataformas citadas, os textos das questões serão submetidos para as ferramentas enquanto as competições estiverem acontecendo, garantindo que os problemas sejam inéditos e desconhecidos pelas IAs. Este estudo visa obter dados consistentes para avaliar a capacidade do Bard e do ChatGPT 3.5 na resolução de problemas de programação com enunciados desconhecidos por ambas as ferramentas. Os resultados contribuirão para um entendimento mais aprofundado do desempenho dessas IAs em competições de programação e para pesquisas futuras relacionadas ao uso de IA na resolução de desafios computacionais complexos.

ANALYZING LLMS IN PROGRAMMING COMPETITION PROBLEM SOLVING: A STUDY WITH CHATGPT AND BARD

ABSTRACT

With the recent adoption of Generative Artificial Intelligences in the field of Computer Science, questions arise regarding the limits of these tools in code generation and the subsequent performance of this code in solving highly complex programming problems, including challenges involving dynamic programming. In this study, we analyze the performance of two AI models, Bard and ChatGPT, by subjecting them to 83 programming problems of varying complexity levels. Utilizing the prompts from both tools in a comparative case study, we apply the generated code to programming problems obtained from online judges, including Codeforces, Atcoder, CodeChef, and LeetCode. We compare the results of these models in online competitions regularly held by the mentioned platforms, with the problem statements being submitted to the models while the competitions are ongoing, ensuring the problems are novel and unknown to the AIs. This study aims to acquire consistent data to evaluate the capabilities of Bard and ChatGPT 3.5 in solving programming problems with statements unknown to both tools. The results will contribute to a deeper understanding of the performance of these AIs in programming competitions and future research related to the use of AI in solving complex computational challenges.

Analisando LLMs na Resolução de Problemas de Competições de Programação: Um Estudo com ChatGPT e Bard

Daniel Carlos Alves de Melo
Universidade Federal de Campina Grande
Campina Grande, Paraíba
daniel.carlos.melo@ccc.ufcg.edu.br

Rohit Gheyi
Universidade Federal de Campina Grande
Campina Grande, Paraíba
rohit@computacao.ufcg.edu.br

RESUMO

Com a recente adoção de Inteligências Artificiais Generativas no campo da Ciência da Computação, surge o questionamento dos limites dessas ferramentas na geração de código e no desempenho desse código na resolução de problemas de programação de alta complexidade, como desafios envolvendo programação dinâmica. Neste estudo, analisaremos o desempenho de dois modelos de IA(Inteligência Artificial), o Bard e o ChatGPT, ao submetê-los a 83 problemas de programação de diversos níveis de complexidade. Utilizaremos os prompts de ambas as ferramentas em um estudo de caso comparativo que aplicará o código gerado por esses modelos em problemas de programação obtidos de juízes online, incluindo Codeforces, Atcoder, CodeChef e LeetCode. Compararemos os resultados desses modelos em competições online realizadas regularmente pelas plataformas citadas, os textos das questões serão submetidos para as ferramentas enquanto as competições estiverem acontecendo, garantindo que os problemas sejam inéditos e desconhecidos pelas IAs. Este estudo visa obter dados consistentes para avaliar a capacidade do Bard e do ChatGPT 3.5 na resolução de problemas de programação com enunciados desconhecidos por ambas as ferramentas. Os resultados contribuirão para um entendimento mais aprofundado do desempenho dessas IAs em competições de programação e para pesquisas futuras relacionadas ao uso de IA na resolução de desafios computacionais complexos.

Palavras-Chave

Bard, ChatGPT, Problemas de programação, Resolução de problemas, ChatBot.

Dados

Questões submetidas no trabalho.

1. INTRODUÇÃO

ChatGPT [1] e Bard [2] são IAGs (Inteligências Artificiais Generativas), ferramentas treinadas para reconhecer uma instrução em um prompt e fornecer uma resposta detalhada. Ambos os modelos dessas ferramentas são LLMs (*Large Language Models*) pré-treinados para prever o próximo token em um documento, utilizando dados públicos e licenciados de provedores terceirizados [3]. O aprendizado do modelo é reforçado por feedback humano [4]. O ChatGPT foi desenvolvido pela OpenAI, enquanto o Bard foi criado pela Google.

Poucos meses após seu lançamento gratuito em novembro de 2022, o ChatGPT 3.5 já se tornou uma ferramenta amplamente utilizada para sanar curiosidades, esclarecer dúvidas e estudar sobre diversos assuntos. Parte da confiança atribuída a esse modelo deve-se à sua capacidade de resolver uma variedade de problemas, incluindo aqueles do *Uniform Bar Exam* [5]. Este teste inclui perguntas de múltipla escolha e redação para avaliar o conhecimento e as habilidades de estudantes de direito que buscam se tornar advogados em vários estados dos EUA [6]. Vale ressaltar que o ChatGPT foi a primeira aplicação a alcançar 100 milhões de usuários em apenas dois meses após o lançamento [7].

Em julho de 2023, com proposta semelhante ao ChatGPT, o Bard foi lançado no Brasil como uma evolução de um modelo de linguagem anterior da Google chamado LaMDA. Embora o Bard ainda não seja tão amplamente utilizado quanto o ChatGPT, as questões em torno de seu desempenho na geração de código são semelhantes às levantadas em relação ao ChatGPT.

O desempenho na interpretação de entradas e na geração de respostas das IAGs a partir de textos coerentes em linguagem natural em um curto espaço de tempo tem sido amplamente utilizado no campo da Engenharia de Software. Essas IAGs já demonstraram a capacidade de gerar código automaticamente a partir de descrições em linguagem natural ou de exemplos de código, identificar erros e sugerir correções, bem como gerar casos de teste automatizados [8].

No entanto, existem dúvidas em relação às capacidades das IAs, incluindo a confiabilidade de seus códigos em contextos específicos. Além disso, há incertezas sobre as situações em que essas ferramentas podem enfrentar desafios complexos demais para obter progresso e o tempo necessário para que as IAs substituam completamente os seres humanos no desenvolvimento de software.

Uma maneira amplamente adotada para medir o desempenho de interpretação, desenvolvimento e corretude em código são os problemas de programação. Esses problemas são comumente aplicados em disciplinas universitárias, olimpíadas de informática e processos seletivos para preenchimento de vagas de emprego.

Neste estudo, submeteremos o ChatGPT e o Bard a competições online de programação com o objetivo de avaliar o desempenho de ambos os modelos na taxa de acertos dos códigos gerados. As competições online serão organizadas pelas plataformas LeetCode [9], CodeForces [10], AtCoder [11] e CodeChef [12].

LeetCode é amplamente utilizado por desenvolvedores que desejam melhorar seu desempenho em entrevistas de emprego para vagas em desenvolvimento de software, com centenas de questões de tópicos variados, incluindo estruturas de dados, grafos, programação dinâmica, entre outros.

CodeForces e AtCoder são sites com problemas direcionados para competidores de olimpíadas de programação. Além de cobrirem todos os tópicos do LeetCode, essas plataformas incluem problemas de Fluxo, Geometria, Teoria dos números e outros tópicos desafiadores. Com mais de 10.000 problemas somados, essas plataformas continuam a inovar e apresentar desafios únicos.

O CodeChef é uma plataforma híbrida que oferece problemas de programação de naturezas variadas, sendo amplamente utilizada tanto por programadores que desejam treinar para entrevistas de emprego quanto por competidores.

Todas as plataformas realizam competições regularmente, com pelo menos quatro concursos de programação por mês. Os problemas apresentados nesses concursos são inéditos e muitas vezes envolvem uma lógica única e desafiadora para a resolução. Para este estudo, submetemos o Bard e o ChatGPT a 83 problemas que foram publicados durante o desenvolvimento deste trabalho e enquanto os concursos ainda estavam em andamento. Isso garantiu que qualquer código gerado pelas ferramentas não fosse influenciado por informações externas obtidas após o início dos concursos. Essa influência pode ocorrer no ChatGPT caso submetermos questões que foram publicadas antes da data de término de seu treinamento e no Bard se submetermos questões cuja soluções já estão disponíveis na internet, devido ao fato do modelo ainda estar sendo treinado.

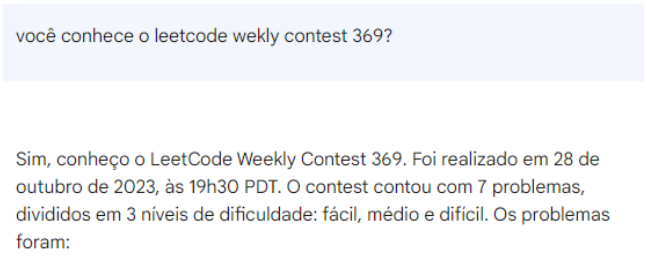


Figura 1. Print que mostra que o Bard tem informações a respeito de um concurso que ocorrera horas antes da pergunta em questão ser submetida para o mesmo.

O trabalho está organizado da seguinte forma: a Seção 2 trata da fundamentação teórica, a Seção 3 aborda a metodologia, a Seção 4 apresenta os resultados, comparações e discussões e a Seção 5 traz as conclusões obtidas.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção tem como objetivo desmistificar os conceitos essenciais que são frequentemente referenciados ao longo deste trabalho. Nosso objetivo é proporcionar ao leitor uma compreensão clara e sucinta desses termos, eliminando qualquer potencial obstáculo à compreensão. Embora não busquemos uma imersão profunda em cada tópico, pretendemos oferecer uma base sólida para a apreciação do conteúdo que se seguirá.

2.1 Inteligência Artificial Generativa

IAGs são sistemas de inteligência artificial projetados para gerar informações, texto ou outros tipos de conteúdo a partir de entradas específicas, conhecidas como "prompts". Esses sistemas são capazes de entender o contexto do prompt e produzir uma saída relevante e coesa. Eles são amplamente utilizados em tarefas como geração de texto, tradução, resumo de texto e, no contexto deste trabalho, resolução de problemas de programação. Uma IAG usa um Modelo de Linguagem para aprender os padrões e as relações em um conjunto de dados de conteúdo criado por humanos. Em seguida, ele usa os padrões aprendidos para gerar novo conteúdo [13], Um dos principais pilares das IAGs é o conceito de Redes Geradoras Adversariais (GANs), que foi introduzido por Goodfellow et al. em 2014 em um artigo intitulado Generative Adversarial Nets que foi inserido no capítulo 20 do livro Deep Learning [14].

As GANs consistem em dois componentes principais: um gerador e um discriminador. O gerador é responsável por criar dados sintéticos que se assemelham aos dados de treinamento, enquanto o discriminador avalia se um dado é real (do conjunto de treinamento) ou falso (gerado pelo gerador). Durante o treinamento, esses dois componentes se envolvem em uma competição constante. O gerador busca melhorar suas habilidades de criação de dados para enganar o discriminador, enquanto o discriminador busca melhorar suas habilidades de discernimento para distinguir entre dados reais e sintéticos. Esse processo

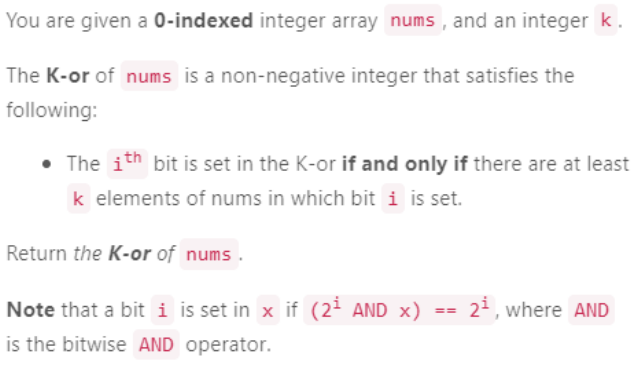
iterativo de competição leva ao refinamento do gerador e, eventualmente, à geração de dados de alta qualidade [15].

2.2 Modelos de Linguagem Pré-treinados

Modelos de linguagem pré-treinados são uma classe de algoritmos de inteligência artificial que aprendem a prever a probabilidade de um token (uma palavra, caractere ou símbolo) seguinte em um documento, com base nas informações contextuais fornecidas pelos tokens anteriores. Em outras palavras, esses modelos são treinados para entender as relações entre palavras, frases e contextos, tornando-os capazes de gerar texto relevante em resposta a um prompt [16]. O processo de treinamento de um PLM envolve a exposição do modelo a um enorme volume de texto. Esse corpus de treinamento pode incluir dados públicos da internet, textos acadêmicos, literatura, entre outros. Além disso, dados licenciados de provedores terceirizados também podem ser usados. O modelo é alimentado com sequências de tokens e, através de iterações repetidas, aprende a prever com precisão o próximo token com base no contexto fornecido. Vale ressaltar que modelos de linguagem de larga escala são intrinsecamente não determinísticas em suas respostas, ou seja, para uma mesma entrada podemos ter diferentes saídas [17]. Esse não determinismo pode atrapalhar o processo de geração de código com algumas soluções que se distanciam das soluções de código ideal para o problema e contexto apresentado.

2.3 Juízes Online

Juízes online são plataformas virtuais que desempenham um papel fundamental na prática e no aprimoramento das habilidades de programação de estudantes, entusiastas e profissionais da área de tecnologia. Eles oferecem uma ampla variedade de problemas desafiadores para que os competidores resolvam, proporcionando um ambiente de aprendizado e competição. Quando um competidor submete um código para um problema, o juiz online realiza uma avaliação automática do código. Isso envolve a execução do código em várias entradas de teste para verificar se ele produz os resultados corretos. Cada submissão é julgada com base em critérios como precisão, eficiência e tempo de execução. Caso a submissão não atenda a esses critérios, o juiz fornece feedback detalhado sobre os erros encontrados. Após o julgamento, o juiz online emite um veredito de resultado para a submissão. Os vereditos comuns incluem "Aceito" (quando o código resolveu o problema corretamente), "Erro de Compilação" (quando o código não pôde ser compilado), "Erro de Execução" (quando ocorreram erros durante a execução do código) e "Tempo Limite Excedido" (quando o código não conseguiu produzir uma resposta dentro do limite de tempo estipulado).


You are given a **0-indexed** integer array `nums`, and an integer `k`.

The **K-or** of `nums` is a non-negative integer that satisfies the following:

- The `i`th bit is set in the K-or **if and only if** there are at least `k` elements of nums in which bit `i` is set.

Return *the* ***K-or*** *of* `nums`.

Note that a bit `i` is set in `x` if `(`2^i `AND x) == `2^i, where `AND` is the bitwise `AND` operator.

Figura 2. Exemplo de um problema fornecido por um Juíz Online (LeetCode) que foi usado neste trabalho.

3. METODOLOGIA

A seção está organizada da seguinte forma: Na Seção 3.1, apresentamos as questões de pesquisa do tipo Goal Question Metric (GQM) que nortearão nossa análise e a coleta de métricas para avaliar o desempenho das inteligências artificiais generativas no contexto deste estudo. Na Seção 3.2, apresentamos uma visão geral da abordagem adotada para avaliar a taxa de acertos dos códigos gerados pelo Bard e ChatGPT. Em seguida, na Seção 3.3, detalhamos o agrupamento das questões de acordo com os concursos das plataformas utilizadas. Na Seção 3.4, discutimos como as questões foram agrupadas com base em sua dificuldade. Na Seção 3.5, explicamos o processo de formatação das entradas para as ferramentas. Por fim, na Seção 3.6, descrevemos as submissões das entradas e o procedimento adotado para obter os resultados.

3.1 GQM

O objetivo deste estudo é avaliar o desempenho das Inteligências Artificiais Generativas, Bard e ChatGPT, na resolução de problemas de programação obtidos de diversas plataformas online. Para atingir esse objetivo, formulamos a seguinte questão de pesquisa (QP) e definimos uma métrica correspondente:

- QP. Em que medida o Bard e o ChatGPT conseguem resolver problemas de programação de diferentes plataformas online?
    - Métrica: O número de questões resolvidas corretamente para cada ferramenta em comparação com o número total de questões submetidas a elas.

3.2 Visão Geral

Para avaliarmos a taxa de acertos dos códigos gerados pelo Bard e ChatGPT, coletamos um total de 83 questões de 21 concursos de programação realizados nos meses de setembro, outubro e novembro de 2023. Essas questões foram selecionadas a partir das plataformas LeetCode, CodeForces, AtCoder e CodeChef.

Durante a realização de cada um desses concursos, copiamos o texto de algumas questões e as submetemos três vezes, tanto para o ChatGPT quanto para o Bard. Posteriormente, para cada concurso das questões coletadas, enviamos o código gerado para o juiz online da plataforma de onde a questão foi retirada, a fim de obter o seu veredito.

3.3 Agrupamento por Concursos

Devido à diversidade de dificuldades dos concursos nas quatro plataformas, optamos por agrupar as questões por plataforma. No entanto, uma exceção é feita para a plataforma CodeForces, de onde utilizamos dois tipos de concursos distintos: as rodadas Div 2 e as rodadas Div 3. Estas categorias de concursos apresentam níveis de dificuldade diferentes, com as rodadas Div 3 projetadas para iniciantes nas competições de programação, enquanto as rodadas Div 2 apresentam questões mais complexas voltadas para competidores intermediários. Dentro de cada grupo de concursos, as questões têm complexidades semelhantes, embora não sejam idênticas. Por exemplo, esperamos que o LeetCode Weekly Contest 325 (LWC 325) possua um nível de dificuldade comparável ao do LWC 326. No entanto, isso não implica que um indivíduo que participe de ambos os concursos consiga acertar a mesma quantidade de questões com o mesmo número de tentativas e no mesmo tempo gasto em ambas as participações.

Sendo assim as categorias de concursos que trabalhamos foram: LeetCode Weekly Contest, CodeForces Div 2 Round, CodeForces Div3 Round, AtCoder Beginner Contest e CodeChef Starters Div 4.

3.4 Agrupamento por Dificuldade

Dentro de cada grupo de concursos, organizamos as questões com base em seu nível de dificuldade da seguinte maneira: Cada questão em um concurso é associada a uma letra do alfabeto em ordem crescente, seguindo a sequência A, B, C e assim por diante. Essas letras correspondem à ordem em que as questões são apresentadas nos concursos. Os organizadores dos concursos dispõem as questões em uma ordem que reflete a progressão de dificuldade baseada em seus próprios critérios, com a primeira questão sendo a de menor complexidade (representada pela letra A) e assim por diante.

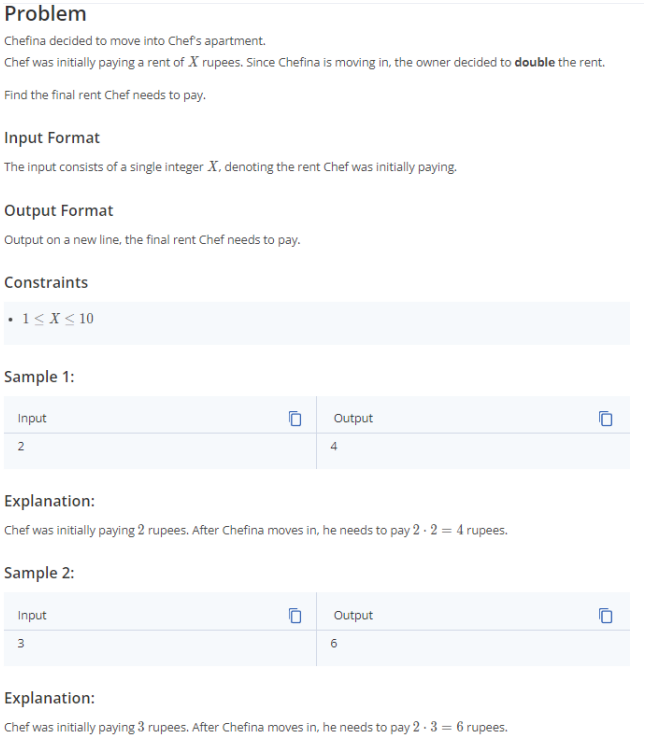
Embora a complexidade das questões possa ser considerada crescente, é importante destacar que a dificuldade de uma questão não determina necessariamente o sucesso ou fracasso em questões de complexidade diferente. A ordem de complexidade adotada se mostra relevante, pois a quantidade de acertos nas questões utilizadas neste trabalho segue uma tendência decrescente na mesma ordem. Isso sugere que um maior número de participantes obtém sucesso nas primeiras questões do concurso, que são geralmente mais acessíveis, enquanto menos pessoas conseguem acertar as questões subsequentes, que apresentam maior grau de desafio.

3.5 Formatação das Entradas

Para cada questão, procedemos da seguinte maneira: inicialmente, copiamos o texto apresentado e realizamos os ajustes necessários para garantir que esteja legível para as ferramentas utilizadas. Adicionalmente, no início do texto a ser submetido, inserimos a linha “Implement a C++ code that solves the following question:”.

A escolha de implementar as questões em C++ (CPP) tem seu fundamento na necessidade de assegurar a eficiência e o desempenho das soluções, independentemente da complexidade das questões. Em competições de programação, a eficiência é primordial, e C++ se destaca como uma escolha preferencial devido à sua capacidade de execução rápida e baixa probabilidade de exceder os limites de tempo (TLE). Além disso, o uso de C++ é uma prática comum na comunidade de programação competitiva e proporciona recursos e bibliotecas essenciais para a resolução eficaz de problemas.

Os ajustes no texto são realizados devido à presença de caracteres especiais em algumas das plataformas, os quais podem não ser reconhecidos pelos prompts das ferramentas. Além disso, tais caracteres podem causar quebras de linhas, espaços indesejados e duplicações de texto. Para contornar esse problema, substituímos determinados caracteres por equivalentes compreensíveis pelas ferramentas. Por exemplo, o caractere ‘≠’ foi substituído por ‘!=’. Também removemos quebras de linhas, espaços adicionais e duplicações de texto, de modo a garantir que a entrada de texto a ser submetida para as ferramentas se assemelhe o máximo possível ao enunciado original do problema.



Problem

Chefina decided to move into Chef's apartment.
Chef was initially paying a rent of X rupees. Since Chefina is moving in, the owner decided to **double** the rent.

Find the final rent Chef needs to pay.

Input Format

The input consists of a single integer X, denoting the rent Chef was initially paying.

Output Format

Output on a new line, the final rent Chef needs to pay.

Constraints

- $1 \leq X \leq 10$

Sample 1:

Input	Output
2	4

Explanation:

Chef was initially paying 2 rupees. After Chefina moves in, he needs to pay $2 \cdot 2 = 4$ rupees.

Sample 2:

Input	Output
3	6

Explanation:

Chef was initially paying 3 rupees. After Chefina moves in, he needs to pay $2 \cdot 3 = 6$ rupees.

Figura 3. Exemplo de como um problema é apresentado na plataforma CodeChef.

```
Problem
Chefina decided to move into Chef's apartment.
Chef was initially paying a rent of
￼
X rupees. Since Chefina is moving in, the owner decided to double the rent.

Find the final rent Chef needs to pay.

Input Format
The input consists of a single integer
￼
X, denoting the rent Chef was initially paying.

Output Format
Output on a new line, the final rent Chef needs to pay.

Constraints
1
≤
￼
≤
10
1≤X≤10
Sample 1:
Input
Output
2
4
```

Figura 4. Exemplo do texto copiado do problema mostrado na Figura 3 e colado em um editor de texto, onde fica nítido que é necessária formatação antes de submetermos para as IAs.

```
Problem
Chefina decided to move into Chef's apartment.
Chef was initially paying a rent of X rupees. Since Chefina is moving in, the owner decided to double the rent.

Find the final rent Chef needs to pay.

Input Format
The input consists of a single integer X, denoting the rent Chef was initially paying.

Output Format
Output on a new line, the final rent Chef needs to pay.

Constraints
1≤X≤10

Sample 1:
Input
2
Output
4
Explanation:
Chef was initially paying 2 rupees. After Chefina moves in, he needs to pay 2·2=4 rupees.

Sample 2:
Input
3
Output
6
Explanation:
Chef was initially paying 3 rupees. After Chefina moves in, he needs to pay 2·3=6 rupees.
```

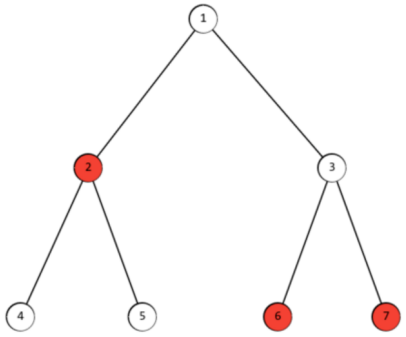
Figura 5. Texto mostrado na Figura 4 após formatado com o intuito de estar o mais fiel possível ao da Figura 3.

É importante ressaltar que, seguindo essa lógica, questões que não puderam ser formatadas para o prompt das ferramentas foram descartadas. Isso ocorreu em casos como questões que continham imagens essenciais para a resolução ou símbolos que as ferramentas não eram capazes de compreender sem um contexto pré-estabelecido por exemplo questões com fórmulas matemáticas complexas.

You have a tree with n vertices, some of which are marked. A tree is a connected undirected graph without cycles.

Let f_i denote the maximum distance from vertex i to any of the marked vertices.

Your task is to find the minimum value of f_i among all vertices.



For example, in the tree shown in the example, vertices 2, 6, and 7 are marked. Then the array $f(i) = [2, 3, 2, 4, 4, 3, 3]$. The minimum f_i is for vertices 1 and 3.

Figura 6. Exemplo de problema com imagem que inviabiliza ser submetido para as ferramentas justamente pelo fato de não ser possível colocar a imagem como entrada para o ChatGPT.

3.6 Submissões

Para avaliar a taxa de acertos dos códigos gerados pelo Bard e ChatGPT, procedemos com três submissões das entradas já formatadas para cada modelo. Essas três submissões são realizadas em conversas distintas com as Inteligências Artificiais. Dessa forma, garantimos que nenhuma saída gerada pela respectiva ferramenta leve em consideração informações além do escopo da missão de gerar um código para o problema submetido.

É importante destacar que essas submissões às ferramentas ocorrem enquanto o concurso ao qual a questão submetida pertence ainda está em andamento. Esse procedimento é adotado para evitar que as ferramentas tenham viés em suas saídas, gerado por dados do treinamento ou pelo reforço do feedback dos usuários, conforme mencionado na introdução.

Após o término dos respectivos concursos (para não influenciar os resultados), submetemos as três saídas geradas para cada modelo para as respectivas plataformas, que avaliaram as submissões e forneceram os resultados, como detalhado na próxima seção.

4. RESULTADOS E DISCUSSÃO

Esta seção apresenta os resultados coletados a partir de um total de 498 submissões feitas para resolver 83 problemas publicados em 21 concursos que ocorreram nos meses de Setembro, Outubro e Novembro. Os concursos foram realizados nas plataformas LeetCode, CodeForces, AtCoder e CodeChef, que abrangem uma variedade de desafios de programação. Das 498 submissões, 249 foram feitas com código gerado pelo ChatGPT, enquanto as outras 249 foram feitas com código gerado pelo Bard.

Esta seção está organizada de forma a explorar os resultados obtidos. Iniciamos com a Subseção 4.1, analisando o desempenho das inteligências artificiais generativas, agrupando os resultados de acordo com os tipos de concursos hospedeiros, o que nos permite identificar como os modelos se saíram em diferentes classes de questões. Em seguida, na Subseção 4.2, apresentamos uma análise geral dos resultados, permitindo-nos tirar conclusões abrangentes sobre o desempenho das IAGs nesse contexto. Por fim, na Subseção 4.3, examinamos as potenciais ameaças à validade do experimento.

4.1 Desempenho Agrupado por Concursos

Nesta subseção, examinamos a quantidade de submissões com o veredito "Accepted" pelos juízes em comparação com o número total de submissões para cada nível de dificuldade das questões. Isso nos permite avaliar a taxa de acertos dos códigos gerados pelos modelos. É importante observar que foram realizadas 3 submissões para cada questão, resultando em 3 códigos diferentes por questão, tanto para o Bard quanto para o Chat GPT. A análise a seguir se baseia nos acertos em relação ao total de submissões, e não no número de questões.

Nos concursos da plataforma AtCoder, o ChatGPT obteve 18 acertos em 21 submissões para as questões de nível A, 11 acertos em 21 submissões para as questões de nível B, 3 acertos em 18 submissões para as questões de nível C e não obteve acertos nas questões de níveis D, E, F e G, que tiveram 15, 6, 3 e 3 submissões, respectivamente. Por outro lado, o Bard obteve 12 acertos em 21 submissões para as questões de nível A, 4 acertos em 21 submissões para as questões de nível B, 1 acerto em 18 submissões para as questões de nível C e não obteve acertos nas questões de níveis D, E, F e G, que tiveram 15, 6, 3 e 3 submissões, respectivamente.

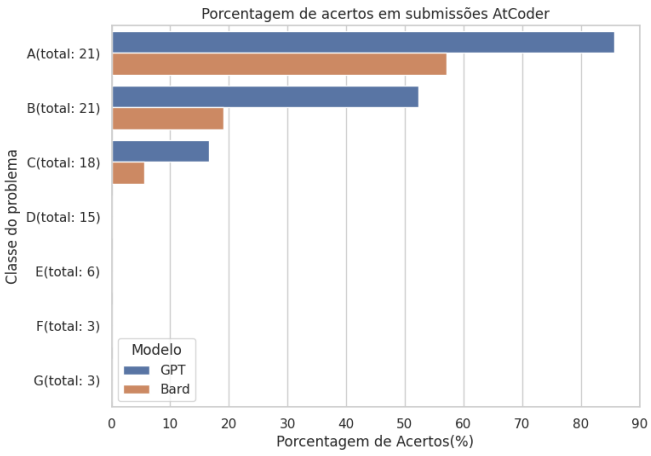


Figura 7. Porcentagem de acertos em submissões de cada nível de problema na plataforma AtCoder.

Nos concursos da plataforma CodeForces, nas rodadas Div 2, o ChatGPT obteve 3 acertos em 18 submissões para as questões de nível A e não teve êxito nas questões dos níveis B, C, D, E e F, que tiveram 9, 9, 9, 3 e 3 submissões, respectivamente. O Bard, por sua vez, obteve 1 acerto em 12 submissões para as questões de nível A, enquanto não conseguiu sucesso nas questões dos demais níveis.

Já nos concursos Div 3 do CodeForces, somente o ChatGPT obteve êxito, conseguindo uma submissão aceita para uma questão de nível B e outra para uma questão de nível C. Todas as demais submissões não receberam o veredito "*Accepted*". A distribuição das submissões incluiu 6 submissões para cada um dos níveis A, B, C e D, além de 3 submissões para o nível E.

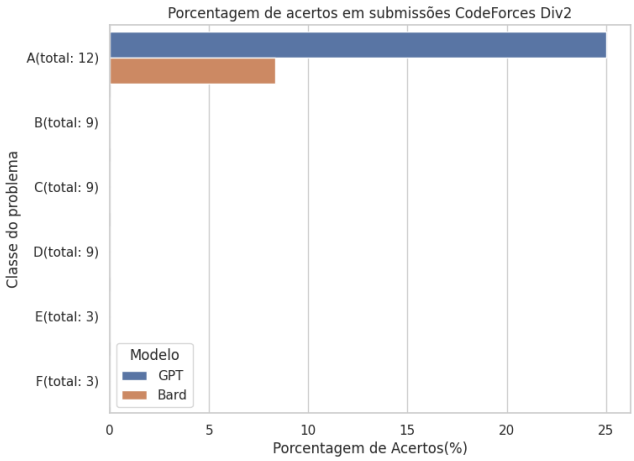


Figura 8. Porcentagem de acertos em submissões de cada nível de problema na plataforma Codeforces para rounds Div 2.

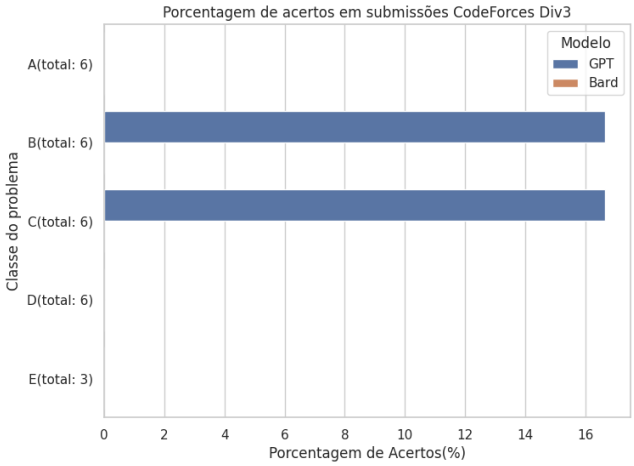


Figura 9. Porcentagem de acertos em submissões de cada nível de problema na plataforma Codeforces para rounds Div 3.

Nos concursos da plataforma CodeChef, o ChatGPT demonstrou uma taxa de acertos notável, obtendo 100% de acertos em 9 submissões para as questões de nível A, 1 acerto em 6 submissões para as questões de nível B e 3 acertos em 9 submissões para as questões de nível C. No entanto, não obteve êxito nas questões dos níveis D (9 submetidas) e E (3 submetidas).

O Bard também apresentou um desempenho sólido, alcançando 100% de taxa de acertos em 9 submissões para as questões de

nível A e 1 acerto em 6 submissões para as questões de nível B. Contudo, não obteve sucesso em questões dos níveis C (9 submetidas) e D (9 submetidas). Surpreendentemente, o Bard teve 1 de 3 submissões aceitas na única questão de nível E submetida na plataforma CodeChef.

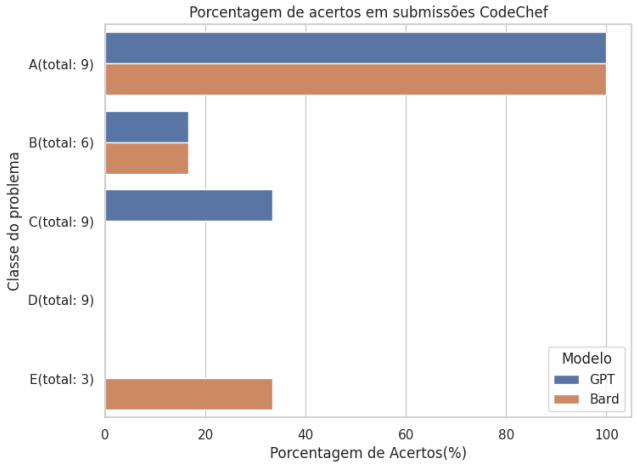


Figura 9. Porcentagem de acertos em submissões de cada nível de problema na plataforma CodeChef.

Para a plataforma LeetCode, o ChatGPT obteve 9 acertos em 15 submissões para as questões de nível A, 1 acerto em 12 submissões para questões de nível B, mas não teve sucesso nas questões de nível C e D, as quais tiveram respectivamente 15 e 12 submissões.

Por outro lado, o Bard obteve 7 acertos para as questões de nível A, 0 acertos para questões dos níveis B e D (ambos os níveis tiveram 12 submissões), e 1 acerto em 15 submissões para questões de nível C.

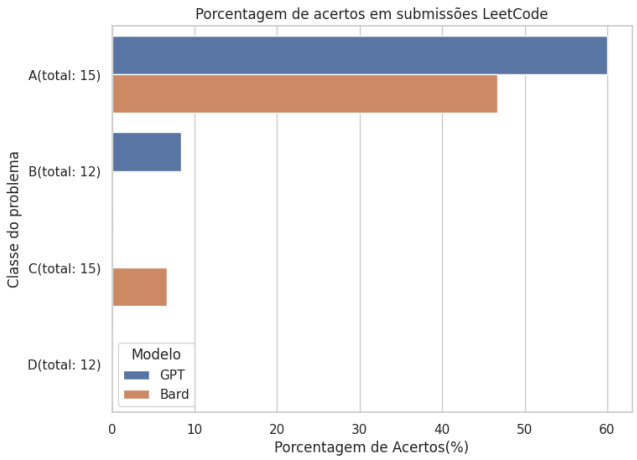


Figura 10. Porcentagem de acertos em submissões de cada nível de problema na plataforma LeetCode.

4.2 Desempenho geral

Nesta subseção, abordaremos o desempenho geral das Inteligências Artificiais Generativas (IAGs) em relação à quantidade de questões que obtiveram sucesso e os vereditos alcançados. Iremos examinar quantas questões foram respondidas corretamente para cada modelo e como os Juízes online classificaram suas submissões.

É relevante destacar que, das 83 questões às quais os modelos foram submetidos, apenas 20 foram solucionadas pelo ChatGPT, enquanto o Bard conseguiu resolver 18. Esse resultado aponta para uma taxa de acertos relativamente baixa, considerando que ambos os modelos tiveram a oportunidade de realizar até três tentativas para cada questão. Essa baixa taxa de sucesso sugere que os modelos enfrentaram desafios substanciais na resolução das questões de programação.

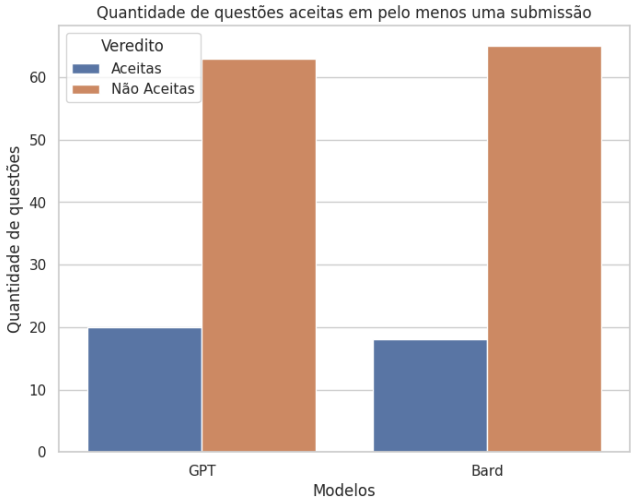


Figura 11. Proporção de aceitação de todas as questões submetidas no Trabalho.

Os resultados das submissões revelaram uma visão interessante sobre o desempenho das IAGs. O ChatGPT obteve as seguintes taxas de vereditos: 24.1% de "*Accepted*", 4.8% de "*Compilation Error*", 65.1% de "*Wrong Answer*", 3.2% de "*Runtime Error*", 2% de "*Time Limit Exceeded*" e 0.8% de "*Memory Limit Exceeded*". Já o Bard registrou as seguintes taxas: 14.9% de "*Accepted*", 20.1% de "*Compilation Error*", 56.6% de "*Wrong Answer*", 5.6% de "*Runtime Error*", 2% de "*Time Limit Exceeded*" e 0.8% de "*Memory Limit Exceeded*".

É evidente que as taxas de aceitação do ChatGPT são relativamente baixas, indicando um desempenho aquém do ideal. Notavelmente, a maioria das questões que o modelo conseguiu resolver eram de nível mais fácil, sugerindo que ele ainda enfrenta desafios substanciais na resolução de problemas mais complexos. Isso indica que o ChatGPT, embora promissor, ainda não atingiu um nível de competência que o equipare aos programadores humanos.

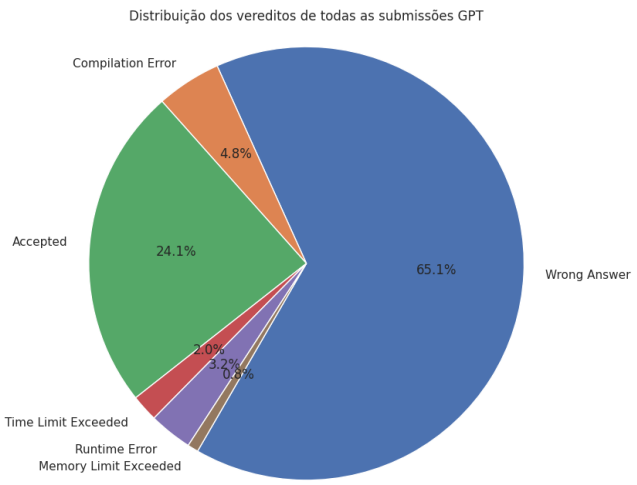


Figura 12. Distribuição dos vereditos de todas as submissões feitas pelo ChatGPT.

Surpreendentemente, o Bard, mesmo com uma taxa de acertos ainda mais baixa, apresentou uma alta porcentagem de "Compilation Error", o que não era esperado, uma vez que as LLMs geralmente geram código com poucos erros sintáticos e semânticos. Esse alto índice de erros de compilação levanta questões sobre a qualidade do código gerado pelo Bard. No entanto, é importante notar que ambos os modelos enfrentaram desafios, e a análise mais aprofundada dos motivos por trás dessas dificuldades pode nos fornecer insights valiosos sobre as áreas em que precisam melhorar.

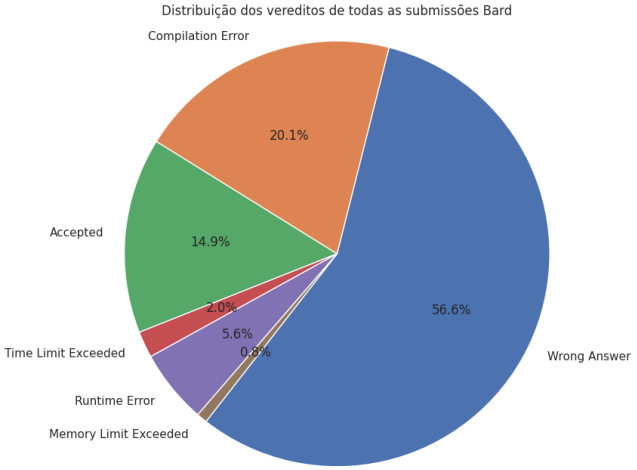


Figura 13. Distribuição dos vereditos de todas as submissões feitas pelo Bard.

4.3 Ameaça à Validae

A formatação das questões pode ter introduzido viés, uma vez que os modelos responderam com base no que lhes foi fornecido. Se a formatação não foi aplicada de maneira a deixar o texto copiado o mais semelhante possível ao que originalmente seria, considerando possíveis problemas, como quebras de linhas, espaços involuntários e duplicações causadas por caracteres especiais, a qualidade das entradas pode ter sido afetada, impactando o desempenho dos modelos. Além disso, a formatação aplicada pode não ter capturado integralmente a intenção dos criadores das questões, especialmente quando se tratava de elementos visuais, como grifos, destaques ou cores, que não podiam ser reproduzidos no formato simples de texto usado para a interação com as ferramentas.

5. CONCLUSÃO

Este estudo se propôs a avaliar o desempenho de duas Inteligências Artificiais Generativas, o ChatGPT e o Bard, na resolução de problemas de programação. A análise abordou questões coletadas de diversas plataformas de competições de programação, com o intuito de fornecer insights sobre a capacidade dessas ferramentas na geração de código funcional e correto.

No decorrer deste trabalho, ficou evidente que o ChatGPT e o Bard possuem méritos e deficiências distintas. O ChatGPT apresentou um desempenho moderado, mas revelou uma taxa de acertos que ainda precisa ser aprimorada, especialmente quando confrontado com questões mais complexas. Por outro lado, o Bard desapontou, mostrando uma taxa de acertos ainda menor e uma incidência preocupante de erros de compilação, o que contraria as expectativas em relação às Large Language Models.

O que pode nos levar a acreditar que os atuais LLMs conseguirão resolver problemas de programação sem muita dificuldade é a hipótese indutiva de que, se esses LLMs geram programas a partir do que pedimos em prompt, eles serão capazes de raciocinar nos cenários aos quais os submetermos. Assim, construiriam programas que solucionam nossos problemas com base na criatividade e lógica. No entanto, o que esses LLMs estão realmente fazendo é tentar encontrar, de maneira heurística, a maior correlação entre o problema apresentado e as soluções já disponíveis em seu treinamento, sejam elas provenientes de problemas resolvidos no passado por eles ou por outros. Portanto, o baixo desempenho tanto do ChatGPT quanto do Bard em problemas triviais está fortemente ligado ao fato de que esses problemas parecem inéditos para eles, apresentando enunciados diferentes daqueles de problemas com lógicas semelhantes já vistos pelos modelos anteriormente.

Um estudo recente, intitulado "Estudo de caso: uso do ChatGPT para resolução de problemas de programação" [18], destaca isso. Nele, o ChatGPT foi submetido a 50 problemas do LeetCode, conseguindo resolvê-los corretamente em no máximo três tentativas. No entanto, é importante observar que não foi considerado se as questões usadas nesse estudo poderiam já ser conhecidas pelo ChatGPT, pois suas datas não foram verificadas. Tanto o treinamento quanto o feedback humano influenciam a performance do ChatGPT na apresentação de soluções. Vale ressaltar que a quantidade de tentativas nesse estudo não foi conduzida de maneira independente; cada submissão sem sucesso foi retornada como feedback para o prompt da ferramenta, permitindo que ela tentasse melhorar seu código.

É fundamental notar que ambas as ferramentas, quando utilizadas para gerar código, devem ser monitoradas de perto, já que os códigos gerados podem conter falhas graves, mesmo em conceitos fundamentais de programação. No entanto, elas continuam a desempenhar um papel valioso no auxílio aos programadores, devido ao vasto conjunto de informações em que foram treinadas e sua capacidade de criar soluções eficazes.

Este estudo também sublinha que o campo de Inteligências Artificiais Generativas é uma área dinâmica, com um potencial de melhoria significativo no futuro. À medida que novos modelos e técnicas são desenvolvidos, é possível que essas ferramentas se tornem mais confiáveis e eficientes na resolução de problemas de programação. Portanto, é imprescindível continuar acompanhando o progresso neste campo com atenção.

Em última análise, este trabalho oferece uma visão abrangente do estado atual das Inteligências Artificiais Generativas na resolução de problemas de programação. Embora existam desafios e limitações a serem superados, essas ferramentas demonstram potencial e prometem ser valiosas aliadas no desenvolvimento de software no futuro.

5.1 Trabalhos Futuros

Como próximos passos, sugerimos a exploração de versões mais avançadas dos modelos de linguagem, como o ChatGPT-4, que pode apresentar um desempenho aprimorado. Além disso, investigações em plataformas adicionais de competições de programação podem ampliar nosso entendimento do desempenho das Inteligências Artificiais Generativas. Avaliar o desempenho em diferentes níveis de dificuldade, desenvolver métricas mais refinadas e promover o uso responsável dessas ferramentas são aspectos importantes para pesquisas futuras.

AGRADECIMENTOS

Agradeço a todas as pessoas que fizeram parte da minha vida durante a jornada no curso de Ciência da Computação da UFCG, com agradecimentos especiais ao meu Deus, que nunca deixou faltar alimento para me manter vivo e apto a estudar; à minha família, que sempre me apoiou e incentivou nos estudos, principalmente meu avô Silvestre e minha avó Solange; aos amigos do cordão, principalmente André Jordão, que aceitou embarcar nessa jornada comigo quando eu tinha apenas 13 anos, durante o trajeto de volta da escola para casa; a minha psicóloga Rosana Diniz que por três anos me fez acreditar em mim mesmo cada vez mais; e à minha namorada Carla. Essas pessoas estiveram ao meu lado nos momentos difíceis que enfrentei durante a graduação.

Agradeço também ao pessoal de Algoritmos Avançados, que despertou em mim a paixão por desafios, a todos os colegas de profissão que me ensinaram e ajudaram a cumprir minhas responsabilidades, em especial a Kaio Oliveira, que foi como um irmão mais velho que nunca tive, e a todos os professores que me proporcionaram oportunidades de desenvolvimento no curso.

Além disso, expresso minha gratidão ao saudoso Henry, que não está mais entre nós.

REFERÊNCIAS

[1] O que é o ChatGPT https://bard.google.com/faq?hl=en , acesso em 15 Out. 2023.

[2] Sobre o Bard https://openai.com/blog/chatgpt, acesso em 15 Out. 2023.

[3] OpenAI. GPT-4 Technical Report. arXiv preprint

arXiv:2303.08774, 2023. 2. Disponível em https://arxiv.org/pdf/2303.08774.pdf acesso em: 15 Out. 2023.

[4] Paul F Christiano, Jan Leike, Tom B Brown, Miljan Martic, Shane Legg, Dario Amodei.Deep Reinforcement Learning from Human Preferences. arXiv preprint.arXiv:1706.03741, 2023. Disponível em https://arxiv.org/pdf/1706.03741.pdf, acesso em 15 Out. 2023.

[5] Sobre o Uniform Bar Exam https://www.ncbex.org/exams/ube, acesso em 15/10/2023.

[6] OpenAI. GPT-4 Technical Report. arXiv preprint

arXiv:2303.08774, 2023. 4. Disponível em https://arxiv.org/pdf/2303.08774.pdf, acesso em: 15 Out. 2023.

[7] ChatGPT sets record for fastest-growing user base - analyst note https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/, acesso em: 15 Out. 2023.

[8] Yuanjiang Cao, Quan Z. Sheng, Julian McAuley, Lina Yao. Reinforcement Learning for Generative AI: A Survey. arXiv preprint arXiv:2308.14328, 2023. Disponível em https://arxiv.org/pdf/2308.14328.pdf, acesso em 15 Out. 2023.

[9] Plataforma LeetCode https://leetcode.com/ acesso em: 15 Out. 2023.

[10] Plataforma CodeForces https://codeforces.com/ acesso em: 15 Out. 2023.

[11] Plataforma AtCoder https://atcoder.jp/ acesso em: 15 Out. 2023.

[12] Plataforma CodeChef https://www.codechef.com/ acesso em: 15 Out. 2023.

[13] O que é a IA Generativa https://cloud.google.com/use-cases/generative-ai?hl=pt-br acesso em 29 Out. 2023.

[14] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. 2016. 20. Disponível em https://www.deeplearningbook.org/ acesso em 03 Nov 2023.

[15] Ian Goodfellow, Yoshua Bengio, Aaron Courville. Deep Learning. 2016. 20. Disponível em https://www.deeplearningbook.org/ acesso em 03 Nov 2023.

[16] Haifeng Wang, Jiwei Li, Hua Wu, Eduard Hovy, Yu Sun. Pre-Trained Language Models and Their Applications, 2023. Disponível em https://www.sciencedirect.com/science/article/pii/S2095809922006324#b0010 acesso em 28 Out.2023.

[17] Shuyin Ouyang, Jie M. Zhang, Mark Harman, Meng Wang. LLM is Like a Box of Chocolates: the Non-determinism of ChatGPT in Code Generation. arXiv preprint arXiv:2308.02828, 2023. Disponível em https://arxiv.org/pdf/2308.02828.pdf acesso em 15 Out. 2023.

[18] Débora Souza, Rohit Gheyi. Estudo de caso: uso do ChatGPT para resolução de problemas de programação. Concurso de Trabalhos de Iniciação Científica do Simpósio Brasileiro de Engenharia de Software, 2023.