Reprodução de Anishka et al.

José Robson da Silva Araujo Junior[1]

[1]Universidade Federal de Campina Grande

Introdução

Com o ChatGPT sendo disponibilizado publicamente pela OpenAI[1], a discussão acerca do potencial de *Large Language Models* (LLMs) no contexto da educação tem se tornado cada vez mais relevante. Alguns trabalhos se dedicam a estudar, por exemplo, a capacidade e aplicabilidade desse sistema como um tutor ou assistente virtual em sala de aula, incluindo na interação direta com alunos [1, 2] e no apoio aos professores e alunos assistentes (*teaching assistants* ou TAs) [3, 4].

O presente trabalho é a reprodução de uma parte do artigo "*Can ChatGPT Play the Role of a Teaching Assistant in an Introductory Programming Course?*" [4]. Em sua pesquisa, Anishka et al. investigam a possibilidade de utilizar o ChatGPT para apoiar tarefas comumente associadas a TAs no contexto de uma disciplina de programação para iniciantes. Em específico, foca-se em duas tarefas: atribuir notas e prover feedback a atividades feitas pelos alunos.

O artigo original conclui que o ChatGPT ainda enfrenta diversas limitações em relação à atribuição de notas, mas que apresenta sugestões de melhoria que, quando implementadas, mostram-se úteis para aperfeiçoar características de qualidade de código.

Na seção seguinte, discute-se o objeto escolhido para a reprodução. Em seguida, resume-se a metodologia do artigo original, a fim de permitir o entendimento da distinção entre as metodologias dos dois trabalhos. Por fim, apresenta-se o resultado da reprodução, as conclusões, as limitações e disponibilizam-se os materiais utilizados.

Objeto da reprodução

Neste trabalho em específico, foca-se na capacidade do ChatGPT na tarefa de atribuir notas a atividades, por ser uma área com seus próprios desafios e por ter resultados que podem ser comparados mais diretamente com os apresentados no artigo.

Como a pesquisa original não disponibilizou os dados utilizados e não foram identificados *datasets* públicos de atividades e correções manuais, decidiu-se recorrer a submissões feitas em uma plataforma de programação competitiva. Similarmente, a pesquisa de Phung et al. [3] utilizou códigos gerados a partir de questões de uma plataforma de estudo de programação para testar o desempenho do ChatGPT.

No caso do Codeforces[2], a plataforma escolhida para este artigo, as submissões estão disponíveis publicamente, incluindo o veredito atribuído pela correção automática, que é utilizada neste trabalho como *ground truth*.

Metodologia original

Nesta seção, apresentam-se, resumidamente, os principais aspectos da metodologia adotada no artigo original. Como o foco deste trabalho é a atribuição de notas, limitou-se a abordar a parte da pesquisa identificada como *Experiment 1: Grading Student Code Submissions*.

Coleta de dados

Os dados utilizados foram originados pelos próprios autores do artigo. O *dataset* consistia em respostas a atividades feitas na linguagem de programação Python dentro de um curso de introdução à programação, incluindo as notas atribuídas por TAs humanos. O conjunto de dados contemplava três atividades divididas em um total de 23 questões.

A pontuação atribuída pelos TAs permitia que créditos parciais fossem conferidos a respostas incompletas ou parcialmente corretas. Nessa correção, apenas a funcionalidade foi considerada, não levando em conta aspectos de qualidade de código.

Dados gerados

A partir de um *prompt* feito ao ChatGPT através de sua API, geraram-se notas para a funcionalidade de cada uma das submissões, atribuindo 2 pontos para questões totalmente corretas, 1 ponto para questões parcialmente corretas e 0 para questões incorretas ou vazias. Além da corretude, também solicitou-se que fosse avaliada a qualidade do código, atribuindo uma nota numa escala de 1 a 5.

Uma vez que os TAs não avaliaram os códigos em relação à qualidade, utilizou-se o pacote Radon[3] de

[1] `https://openai.com/`

[2] `https://codeforces.com`

[3] `https://pypi.org/project/radon/`

Python para calcular métricas relacionadas à qualidade: o esforço (*Halstead's effort score*, que diz respeito à complexidade do código) e modularidade (que impacta, dentre outras características, a legibilidade do código). Dessa forma, foi possível obter parâmetros de comparação com a nota de qualidade de código atribuída pelo ChatGPT.

Análise de dados

A etapa de análise dos dados consistiu na geração e discussão dos resultados quantitativos: os desvios médio e máximo para as notas de funcionalidade geradas pelo ChatGPT às atividades, a correlação entre essas notas com as que foram atribuídas pelos TAs, e a correlação entre as métricas de qualidade de código e as notas de qualidade geradas pelo ChatGPT.

Diferenças metodológicas com o estudo original

Nesta seção, apresenta-se uma visão geral das diferenças entre metodologia do trabalho original e esta reprodução. Alguns aspectos dessas mudanças são aprofundados na seção posterior.

População estudada

No artigo original Dados relativos a atividades em um curso de introdução à programação, contendo os códigos de alunos e as notas atribuídas por TAs humanos.

Na reprodução Como os dados utilizados na pesquisa original não foram disponibilizados, gerou-se um novo *dataset* a partir de submissões públicas feitas no Codeforces.

Pergunta de pesquisa

No artigo original "*In terms of grading the student code submissions, how does ChatGPT's evaluation compare with the evaluation done by human TAs?*"

Na reprodução Pergunta adaptada, uma vez que as notas não foram atribuídas por TAs humanos. Em termos de atribuir notas a submissões de código, como a avaliação do ChatGPT se compara à avaliação feita pelo Codeforces?

Metodologia de coleta de dados

No artigo original A coleta de dados não é descrita no documento, mas provavelmente foi extraída a partir de alguma ferramenta de atribuição de notas utilizada na instituição em questão.

Na reprodução Extração de submissões a partir das páginas no Codeforces.

Dados gerados

No artigo original Notas para a corretude e qualidade do código atribuídas pelo ChatGPT, além de métricas de qualidade geradas através do pacote Radon de Python.

Na reprodução Similares aos do artigo original, mas sem a possibilidade de notas parciais na corretude.

Metodologia de análise de dados

No artigo original A análise foi feita a partir dos desvios médio e máximo para as notas de corretude em cada atividade, e a correlação entre a atribuição de notas de corretude e de qualidade.

Na reprodução Além das correlações das notas, observou-se a matriz de confusão de atribuição de notas. Utiliza-se, também, inferência estatística para estimar os valores na população.

Código da análise de dados

No artigo original Não disponibilizado.

Na reprodução Gerado pelo autor da reprodução, baseando-se na metodologia original, e disponibilizado publicamente no GitHub.

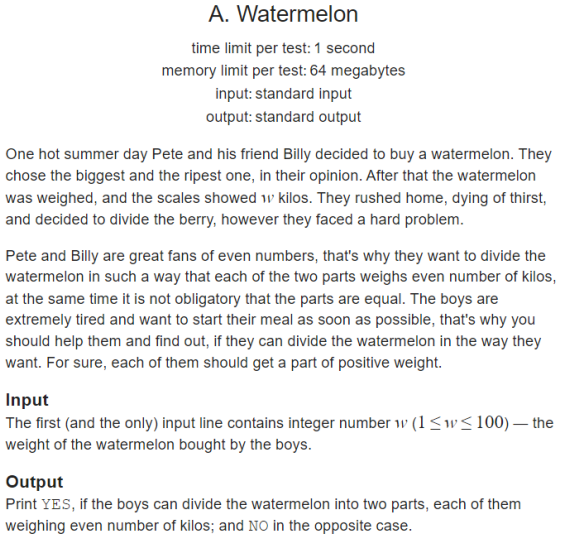
Metodologia da reprodução

Coleta de dados

Uma vez que o *dataset* utilizado pelos pesquisadores no estudo original não foi publicizado, identificou-se o Codeforces como uma possibilidade para a extração de submissões de código. Trata-se de uma plataforma em que estudantes e entusiastas podem participar de *contests* (competições) de programação competitiva e praticar seus conhecimentos com problemas anteriores.

Como se trata de uma plataforma utilizada por iniciantes e proficientes em programação competitiva, a dificuldade dos problemas varia bastante, e é identificada através de *tags*, que também identificam as categorias em que as questões se encaixam (por exemplo, *number theory* e *brute force*). Considerando que o estudo original está inserido no contexto de introdução à programação (incluindo tópicos como algoritmos e estruturas de dados mais simples), utilizou-se um subconjunto de problemas da menor dificuldade disponível no site, identificada pela *tag *800*.

Utilizou-se um *dataset*[4] de problemas do Codeforces para selecionar alguns para análise. Esse conjunto de dados contém os identificadores de cada problema, o seu enunciado e as *tags*. O subconjunto utilizado, contendo 15 questões (todas com a *tag *800*) foi salvo em um arquivo `problems.csv`. Cada questão especifica o problema e expectativas de entrada e saída, como exemplificado pela Figura 1.



A. Watermelon

time limit per test: 1 second
memory limit per test: 64 megabytes
input: standard input
output: standard output

One hot summer day Pete and his friend Billy decided to buy a watermelon. They chose the biggest and the ripest one, in their opinion. After that the watermelon was weighed, and the scales showed w kilos. They rushed home, dying of thirst, and decided to divide the berry, however they faced a hard problem.

Pete and Billy are great fans of even numbers, that's why they want to divide the watermelon in such a way that each of the two parts weighs even number of kilos, at the same time it is not obligatory that the parts are equal. The boys are extremely tired and want to start their meal as soon as possible, that's why you should help them and find out, if they can divide the watermelon in the way they want. For sure, each of them should get a part of positive weight.

Input
The first (and the only) input line contains integer number w ($1 \le w \le 100$) — the weight of the watermelon bought by the boys.

Output
Print `YES`, if the boys can divide the watermelon into two parts, each of them weighing even number of kilos; and `NO` in the opposite case.

Figura 1: *Exemplo de questão do Codeforces (4A, Watermelon)*[5].

Como não foi identificado um *dataset* público de submissões, os códigos foram extraídos diretamente da plataforma Codeforces. O julgador da plataforma utiliza um conjunto de testes para cada questão a fim de avaliar se o código submetido retorna a resposta esperada para os casos de entrada especificados, atribuindo, dessa forma, um veredito. De cada questão, foram extraídas cinco submissões aleatórias julgadas como *Accepted* e cinco julgadas como *Wrong Answer*[6], totalizando dez por questão e 150 ao todo.

Neste trabalho, as questões julgadas como *Accepted* são consideradas corretas (nota 1) e as como *Wrong Answer*, incorretas (nota 0). Utilizou-se uma distribuição igual de submissões corretas e incorretas para evitar um desbalanceamento na análise e para que fosse possível ter uma visão geral dos possíveis erros do julgamento do ChatGPT. Também não se utilizou a noção de créditos parciais, uma vez que muitas vezes os códigos julgados como *Wrong Answer* podem estar parcialmente corretos. As submissões coletadas foram armazenadas em um arquivo `submissions.csv`.

[4] `https://www.kaggle.com/datasets/immortal3/codeforces-dataset`

[6] Há diversos outros vereditos possíveis para questões não julgadas como *Accepted* (como *Time Limit Exceeded* e *Runtime Error*), mas decidiu-se focar nesses dois resultados, uma vez que essas outras conclusões de julgamento nem sempre estão associadas à corretude. Por exemplo: um código pode ter exceder o limite de tempo especificado, mas ainda ser uma solução válida.

Experimento

Inicialmente, seguindo a linha da pesquisa original, utilizou-se um script Python com o uso do pacote Radon para gerar duas métricas relacionadas à qualidade do código para cada submissão: esforço e modularidade.

Nesta reprodução, utilizou-se a versão *flagship* ("carro-chefe") mais recente do ChatGPT, o GPT-4o, promovido pela OpenAI como uma versão mais eficiente e precisa que a versão 3.5 (utilizada no artigo original) e com um conhecimento próximo ao da versão GPT-4, mas com custo reduzido[7]. Uma funcionalidade particularmente útil da versão GPT-4o é a possibilidade de submissão e processamento de arquivos. Dessa forma, foi possível utilizar o cliente web do ChatGPT no lugar de sua API.

Dessa forma, GPT-4o foi alimentado com os arquivos `problems.csv` e `submissions.csv`, e orientado quanto à natureza desses arquivos. Para a geração das notas propriamente ditas, utilizou-se um *prompt* adaptado a partir do que foi apresentado no estudo original, fazendo pequenas alterações para o contexto, de forma a manter uma certa consistência. A nova versão do *prompt* é apresentada na Figura 2.

> You are a teaching assistant for an Introduction To Programming Course, and your task is to grade student code submissions using the provided rubric. The code is written in Python programming language.
>
> You are supposed to mark each code available in the `submissions.csv` file based on functionality, referring back to its corresponding problem statement which is available in the `problems.csv` file. If the code is completely correct, give 1 mark. If the code is incorrect, give 0 marks. If the code is empty, give 0 marks. Do not return the distribution of marks.
>
> In addition to the functionality score, also give a quality score on a scale of 1-5 based on Halstead metrics and code modularity. Do not correct the previous solution or solve the question. Follow all the instructions carefully.
>
> The output should be in the format of a CSV file, containing the columns `submission_id, functionality_score, quality_score`.

Figura 2: *Prompt utilizado na reprodução.*

[7] `https://platform.openai.com/docs/models`

Resultado da reprodução

A distribuição de valores de esforço calculados através do pacote Radon está apresentada na Figura 3. Como essa métrica cresce ilimitadamente conforme há mais código escrito, seu valor não segue uma escala definida. Dessa forma, enquanto a maioria dos códigos teve uma métrica de esforço calculada dentro do intervalo entre 0 e 2000, há um *outlier* com valor próximo a 10000. Considerando que há poucos dados, esse *outlier* tem potencial de interferir nas análises (e de fato isso foi percebido experimentalmente), então decidiu-se desconsiderá-lo quando tratando de esforço.

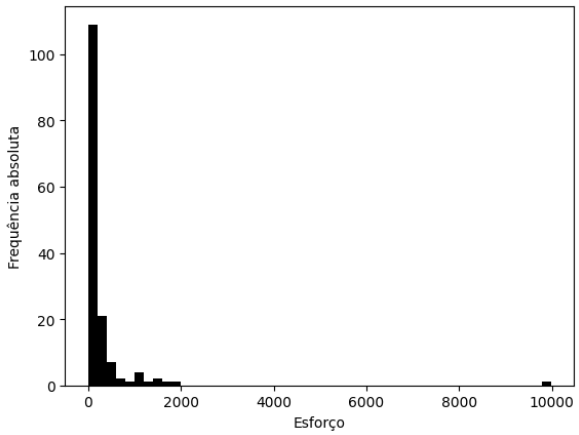


Figura 3: *Histograma de esforço.*

Por sua vez, a distribuição dos valores de modularidade está descrita na Figura 4. O valor dessa métrica está limitado ao intervalo $[0, 100]$; no caso dos dados observados, houve valores entre 40 e 100, com a maior parte concentrada entre 60 e 80, indicando que os códigos têm, em geral, uma boa modularidade. Uma explicação possível para essas notas é o fato de terem sido analisadas soluções para problemas simples e, em muitos casos, com soluções curtas.

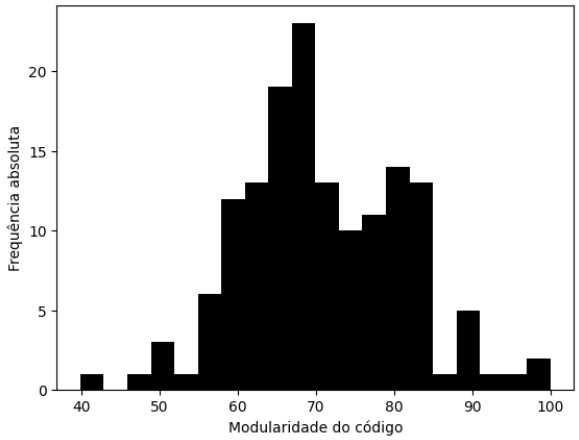


Figura 4: *Histograma de modularidade.*

Observou-se, ainda, a relação entre as métricas de esforço e modularidade. A Figura 5 exibe a dispersão dos valores de esforço (na escala logarítmica) e de modularidade, ilustrando uma aparente relação moderada e negativa entre as variáveis, suportada pelo valor encontrado para o coeficiente de correlação de Spearman: -0.580. Dessa forma, valores mais altos de modularidade parecem estar associados a valores menores de esforço, e vice-versa.

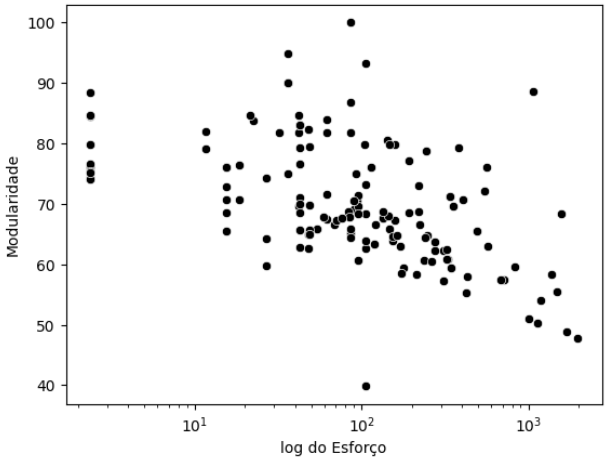


Figura 5: *Relação entre esforço (na escala log) e modularidade.*

Em relação ao conjunto de dados sob análise, o ChatGPT apresentou uma concordância de 66% na atribuição de notas com relação ao julgamento da questão no Codeforces. A Tabela 1 apresenta os resultados por questão, que variaram entre 40% e 90% de concordância.

Tabela 1: *Concordância na avaliação de cada questão.*

Problema	% de Concordância
4A	80%
32A	60%
59A	80%
104A	50%
146A	90%
233A	80%
268A	50%
386A	40%
401A	60%
448A	60%
467A	70%
492A	60%
520A	70%
599A	50%
624A	90%
Média	**66%**

A matriz de confusão, apresentada na Figura 6, destaca que as questões corretas foram comumente

julgadas pelo ChatGPT como corretas; houve poucos falsos negativos. Entretanto, também se percebe que a maior parte das observações foi julgada como correta: 116 (77,33% dos dados). Isso pode indicar que o ChatGPT pode estar mais propenso a julgar códigos como corretos, ou talvez não utilizar métodos que avaliem a corretude adequadamente.

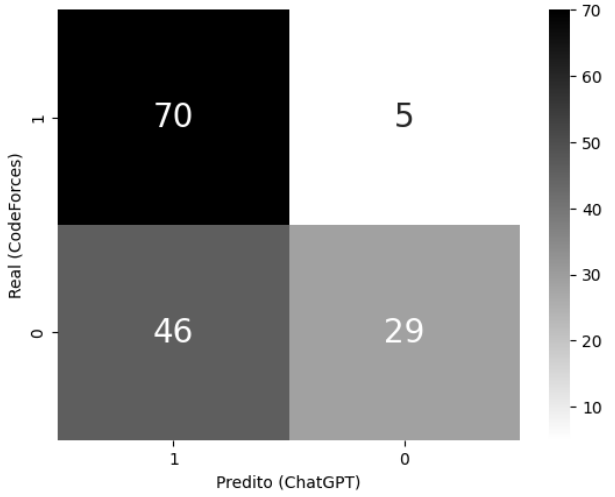


Figura 6: *Matriz de confusão: valores reais são associados ao veredito do código na plataforma Codeforces, enquanto os valores preditos são os apresentados pelo GPT-4o.*

Em relação à qualidade, as notas atribuídas pelo ChatGPT ficaram em um intervalo de 3 a 5, com 132 (88%) dos códigos sendo avaliados com a nota 4.

A fim de propor uma interpretação estatística e expandir a análise para além da amostra, utilizou-se inferência estatística para estimar os valores de coeficiente de correlação entre variáveis de interesse. Os intervalos de confiança (ICs) apresentados na Tabela 2 foram calculados através do processo de *bootstrap* e estimam os coeficientes de *Spearman*.

Tabela 2: *ICs das correlações (ρ) entre métricas calculadas.*

Métricas	**IC (95%)**	**ρ na amostra**
Funcionalidade e veredito	[0.242, 0.505]	0.382
Esforço e modularidade	[-0.693, -0.461]	-0.580
Qualidade e esforço	[-0.289, 0.157]	0.054
Qualidade e modularidade	[-0.217, 0.247]	0.009

A análise desses resultados parece indicar que:

- a correlação entre a nota atribuída pela funcionalidade e o veredito da questão no Codeforces é positiva, podendo ser de fraca a moderada (na amostra em questão, é moderada);
- a correlação entre a nota de modularidade e a métrica de esforço é negativa, podendo ser de moderada a forte (na amostra, é moderada);
- não há correlação significativa entre a nota de qualidade e o esforço, nem entre a nota de qualidade e a modularidade, tendo em vista que ambos os ICs incluem 0 e, mesmo considerando os limites inferior e superior, tratam-se de valores pequenos e próximos a 0.

Conclusões

Os resultados encontrados nesta reprodução parecem corroborar os do estudo original: o ChatGPT, em seu estado atual, não é uma ferramenta necessariamente adequada para a atribuição de notas a exercícios de programação. Apesar disso, através da experimentação, percebeu-se que os resultados para essa tarefa podem ser melhorados de maneira drástica iterativamente, com esclarecimentos feitos em *prompts* adicionais e melhores, o que também é considerado nas conclusões do artigo original.

Para a análise feita aqui, no entanto, utilizou-se a análise inicial, com a finalidade de manter a consistência com o trabalho sendo reproduzido. Na amostra, a correlação entre as notas atribuídas e as notas esperadas tem um valor moderado e positivo, assim como no trabalho original, e o resultado da correlação nula ou inexistente entre as métricas de qualidade com a nota de qualidade também segue do trabalho original.

Diferentemente do artigo original, no entanto, utilizou-se inferência estatística para trazer mais confiança para os resultados apresentados, além de uma métrica de concordância/acurácia para a atribuição de notas, que indicou que o ChatGPT trouxe resultados superiores a uma atribuição aleatória de notas.

Além disso, como foi proposta uma análise a nível de questões corretas e incorretas, foi possível propor novas hipóteses que podem ser investigadas em trabalhos futuros, como a possibilidade de o ChatGPT estar mais propenso a julgar códigos como corretos. Pode-se tornar necessário, assim, gerar *prompts* que detalhem o procedimento de correção com mais detalhes.

Limitações

As limitações deste trabalho incluem o *dataset* utilizado, por ser reduzido e ter sido gerado manualmente. Ainda, não se considerou a possibilidade de atribuição de créditos parciais, o que certamente adiciona uma camada de complexidade para a tarefa de atribuição de notas.

Por fim, questões de plataforma como o Codeforces têm especificações indiretas, enquanto problemas de introdução à programação costumam ser mais simples e diretos, como o exemplo trazido no artigo original. Para tentar contornar esse problema, foram utilizados alguns dos problemas considerados mais "fáceis", mas essa característica ainda pode ter influenciado no resultado.

Material da reprodução

Esta reprodução busca, à sua maneira, propor outras reproduções do trabalho original com as futuras versões do ChatGPT ou mesmo outros LLMs, e utilizando dados diferentes. Para facilitar o processo e possibilitar replicações deste trabalho, os códigos e *datasets* gerados e mencionados neste documento estão disponíveis no GitHub[8].

Referências

[1] Karan Taneja et al. "Jill Watson: A Virtual Teaching Assistant Powered by ChatGPT". Em: *Artificial Intelligence in Education*. Ed. por Andrew M. Olney et al. Cham: Springer Nature Switzerland, 2024, pp. 324–337. ISBN: 978-3-031-64302-6.

[2] Milić Vukojičić e Jovana Krstić. "ChatGPT in programming education: ChatGPT as a programming assistant". Em: *InspirED Teachers' Voice* 2023.1 (2023), pp. 7–13.

[3] Tung Phung et al. "Generative AI for Programming Education: Benchmarking ChatGPT, GPT-4, and Human Tutors". Em: *Proceedings of the 2023 ACM Conference on International Computing Education Research - Volume 2*. ICER '23. Chicago, IL, USA: Association for Computing Machinery, 2023, pp. 41–42. ISBN: 9781450399753. DOI: 10.1145/3568812.3603476. URL: https://doi.org/10.1145/3568812.3603476.

[4] Anishka et al. *Can ChatGPT Play the Role of a Teaching Assistant in an Introductory Programming Course?* 2024. arXiv: 2312.07343 [cs.HC]. URL: https://arxiv.org/abs/2312.07343.

[8] https://github.com/JRobsonJr/fpcc2-reproducao