Uso de LLMs para correção de atividades de programação: uma análise comparativa no contexto de orientação a objetos

Marcel da Silva Melo[1], **Fernando Barbosa Matos**[1], **Rodrigo Elias Francisco**[1], **Cleon Xavier Pereira Júnior** [2], **Rafael Dias Araújo**[3]

[1]Núcleo de Computação – Instituto Federal Goiano – campus Morrinhos
Morrinhos – GO – Brasil

[2]Instituto Federal Goiano – campus Iporá
Iporá – GO – Brasil

[3]Faculdade de Computação – Universidade Federal de Uberlândia (UFU)
Uberlândia – MG – Brasil

{marcel.melo,fernando.matos, rodrigo.francisco}@ifgoiano.edu.br
cleon.junior@ifgoiano.edu.br, rafael.araujo@ufu.br

Abstract. *Grading of programming assignments is a labor-intensive task that requires considerable time and effort from instructors, especially in large classes, which results in a high volume of corrections. In this context, this study aimed to investigate the potential of LLMs, in their default configuration and using a zero-shot approach, for the automatic assessment of Object-Oriented Programming (OOP) activities. A comparative analysis was carried out, using statistical metrics, between the scores generated by the LLMs and the grade assigned by the course instructor. The study indicated that the models GPT-4.1, GPT-4.1-mini, Grok-3, DeepSeek-V3, and Grok-3-mini achieved low error rates and maintained a high level of agreement with the grades assigned by the instructor.*

Resumo. *A correção de atividades de programação é uma tarefa trabalhosa que exige tempo e esforço consideráveis dos docentes, sobretudo em turmas numerosas, o que gera um alto volume de correções. Diante deste cenário, este trabalho visou investigar o potencial de LLMs, em configuração padrão e abordagem zero-shot, na avaliação automática de atividades de Programação Orientada a Objetos (POO). Foi realizada uma análise comparativa, utilizando métricas estatísticas, entre as notas geradas por LLMs e a nota atribuída pelo professor da disciplina. O estudo apontou que os modelos GPT-4.1, GPT-4.1-mini, Grok-3, DeepSeek-V3 e Grok-3-mini alcançaram baixos índices de erros e mantiveram um forte nível de concordância com as notas atribuídas pelo professor.*

1. Introdução

A inteligência artificial (IA) tem passado por uma grande evolução e os Modelos de Linguagem de Grande Escala (do inglês, *Large Language Models* – LLMs) têm de-

monstrado capacidades notáveis de compreensão e geração de texto por meio de algoritmos avançados de processamento de linguagem natural (PLN) [Kasneci et al. 2023, Xie et al. 2024]. Com isso, esses modelos têm se tornado objeto de estudos em diversos campos de pesquisa, entre eles a educação [Marques and Morandini 2024, Montenegro-Rueda et al. 2023, Razafinirina et al. 2024]. Por exemplo, para as práticas docentes, os LLMs permitem a simplificação de tarefas como a elaboração e personalização de materiais didáticos [Razafinirina et al. 2024], a geração automática de questões [Ta et al. 2023, Quincozes et al. 2025], além de correção e geração de *feedback* personalizado em atividades realizadas por estudantes [Seo et al. 2025, Silva and Costa 2025].

Dentre os caminhos possíveis do uso de LLMs, destaca-se o campo de avaliação automática de respostas para atribuição de notas. Pesquisas que dizem respeito a atribuição automática de notas se dividem em estudos para análise de respostas curtas [Ferreira Mello et al. 2025], redações [Lobo et al. 2025] ou até mesmo questões matemáticas [Nakamoto et al. 2023]. No contexto do ensino de programação de computadores, a correção de atividades de maneira manual é uma tarefa trabalhosa e desafiadora que exige tempo e esforço consideráveis de docentes. Essa atividade espera que o professor avalie a adequação da solução ao enunciado, compreenda o raciocínio do estudante, identifique erros e ofereça orientações que promovam a reflexão e a aprendizagem [Silva and Costa 2025, Xie et al. 2024, Yousef et al. 2025].

Assim como LLMs estão sendo investigados para correção automática de redações, respostas curtas e questões matemáticas, estes podem ser utilizados para automatizar a correção de atividades práticas de programação, podendo resultar na redução da carga de trabalho dos professores [Razafinirina et al. 2024, Silva and Costa 2025, Yousef et al. 2025]. No entanto, a maioria desses trabalhos se concentra no uso de LLMs na avaliação automática de trabalhos práticos em disciplinas introdutórias de programação [Balse et al. 2023, Grandel et al. 2024, Silva and Costa 2025, Yousef et al. 2025].

Diferente das disciplinas introdutórias, que normalmente utilizam programação estruturada ou procedural, onde o foco reside na lógica sequencial e na manipulação direta de dados através de variáveis e funções, o ensino de Programação Orientada a Objetos (POO) apresenta desafios adicionais tanto para docentes quanto para os estudantes. A natureza abstrata dos conceitos de POO, combinada com a dificuldade de avaliar adequadamente as atividades dos estudantes e fornecer *feedback* adequado em larga escala, representa uma das dificuldades na educação em ciência da computação [Menolli and Strik 2025, Gutiérrez et al. 2022, Efan et al. 2023]. Toda essa complexidade adicional potencializa as dificuldades da avaliação de atividades práticas tanto de forma manual pelos docentes quanto de forma automática por ferramentas computacionais [Efan et al. 2023].

Devido a complexidade que envolve o ensino/aprendizagem de POO e a necessidade de buscar alternativas que viabilizem o trabalho docente, este estudo tem como objetivo avaliar o comportamento de LLMs aplicados à avaliação automática de atividades práticas de POO, a partir de uma comparação quantitativa com notas atribuídas pelo docente. O trabalho utilizou modelos da família GPT, por frequentemente aparecer em trabalhos prévios [Ferreira Mello et al. 2025], além de outros modelos abertos e proprietários, comumente usados por usuários finais. Ainda nesse aspecto, os modelos foram

testados sem alteração de parâmetros de inferência como *temperature* ou *top-p*, utilizando os modelos tal como são disponibilizados em suas respectivas plataformas oficiais. Essa escolha se justifica pois muitos professores que recorrem aos LLMs para gerar correções o fazem por meio de suas plataformas oficiais e não sabem ou não tendem a realizar tais ajustes técnicos nestes parâmetros, o que torna relevante compreender o comportamento e desempenho destes modelos no cenário comum de uso.

2. Revisão da Literatura

2.1. Desafios no ensino e aprendizagem de POO

O conteúdo de POO é um desafio e, portanto, foco de várias pesquisas e revisões sistemáticas [Gutiérrez et al. 2022, Efan et al. 2023, Menolli and Strik 2025]. Os desafios de aprendizagem apontados nos trabalhos englobam desde problemas na aplicação de conceitos básicos de programação e entendimento da relação entre classe e objeto, até dificuldades com conceitos avançados, como herança, polimorfismo e reutilização de código [Gutiérrez et al. 2022, Efan et al. 2023]. Essa complexidade teórica e o alto nível de abstração de conceitos fundamentais do POO dificulta o aprendizado por parte de estudantes e torna o ensino, avaliação e *feedback* adequado por parte dos docentes um desafio [Efan et al. 2023].

Diferente de atividades práticas da programação procedimental, onde a corretude funcional pode ser um critério primário para avaliação, no paradigma orientado a objetos, devido sua natureza abstrata e complexa, a avaliação deve considerar não apenas se o estudante chegou ao resultado correto, mas como ele estruturou sua solução e se ele de fato compreendeu os princípios da orientação a objetos envolvidos [Efan et al. 2023, Efan et al. 2023]. Essa complexidade adicional demanda ainda mais tempo e esforço do professor no processo de ensino, avaliação e *feedback*, aumentando a importância de uma solução automatizada que alivie essa carga de trabalho docente.

Diante dessas questões, a literatura recente tem explorado soluções para otimizar o processo de avaliação em POO. Uma vertente é o desenvolvimento de ferramentas de avaliação automatizada específicas para POO, que vão além dos testes de saída tradicionais [Menolli and Strik 2025, Martins 2025]. O Java-Judge-OO é uma ferramenta educacional para avaliação automática de códigos Java orientados a objetos com base em uma rubrica estruturada, fornecendo *feedback* consistente e objetivo. Sua principal motivação é reduzir a subjetividade na análise de critérios como definição de classes, encapsulamento e uso adequado de conceitos de POO (abstração, herança e polimorfismo) [Martins 2025].

O trabalho de Strik (2025) apresenta uma abordagem com IA generativa para identificar dificuldades de aprendizagem em POO a partir do código dos estudantes, relacionando *code smells* e violações de princípios da Orientação a Objetos a possíveis problemas conceituais. Para isso, o autor propõe o CodeBuddy, um agente web e/ou integrado ao Visual Studio Code que analisa automaticamente o código, identifica problemas de qualidade em POO e gera *feedback* educacional personalizado [Strik 2025, Menolli and Strik 2025]. Um passo prévio à definição dessas ferramentas de avaliação automatizadas consiste na análise e seleção do LLM mais adequado para a avaliação de código orientado a objetos.

2.2. Avaliação Automática na era de LLM

O uso de LLMs em atividades de ensino de programação tem ganhado destaque nos últimos anos onde vários estudos focam na avaliação automatizada e geração de *feedback* de larga escala, normalmente aplicados em disciplinas introdutórias [Yousef et al. 2025, Balse et al. 2023, Silva and Costa 2025, Grandel et al. 2024].

Balse et al. (2023) mostram que modelos como GPT-3 e GPT-4 conseguem identificar erros e sugerir melhorias com taxas de acerto entre 60% e 70% na correção de atividades de disciplinas introdutórias. Silva e Costa (2025) , ao analisarem quatro LLMs de última geração (GPT-4o, GPT-4o-mini, GPT-4-Turbo e Gemini-1.5-pro), identificaram que, na correção de atividades introdutórias, 63% dos *feedbacks* foram precisos e completos, enquanto 37% apresentaram problemas como alucinações ou localização inadequada de erros, evidenciando riscos pedagógicos e a necessidade de refinamentos.

Akyash et al. (2025) propõem o StepGrade, um *framework* baseado em GPT-4 com *chain-of-thought prompting* para avaliar, de forma contextualizada, funcionalidade, qualidade de código e eficiência em 30 atividades em Python, mostrando maior alinhamento com a avaliação humana do que *prompts* tradicionais. Já Yousef et al. (2025) apresentam o BeGrading, um modelo ajustado (*fine-tuned*) a partir de um grande conjunto de dados reais e sintéticos de tarefas de programação, buscando reduzir a carga docente e fornecer notas e *feedback* consistentes. Os autores relatam diferenças médias em torno de ±0,95 em escala de 0 a 5 quando comparado a modelos de referência, indicando boa confiabilidade para uso prático.

No entanto, em contextos mais avançados, como programação concorrente, o uso de LLMs para correção de atividades pode ainda não trazer resultados satisfatórios. Em seu estudo, Estévez-Ayres et al. (2024) encontraram cerca de 50% de acurácia na detecção de erros complexos (como condições de corrida e *deadlocks*), reforçando a importância da revisão humana e do ajuste fino de *prompts* para alinhar o uso de LLMs a objetivos de aprendizagem mais sofisticados. Já no trabalho de Grandel et al. (2024) , os autores descrevem a ferramenta GreAIter, que utiliza o ChatGPT-4 com uma rubrica estruturada para apoiar a avaliação de atividades avançadas de programação paralela funcional em Java, em um esquema semi-automatizado com *human-in-the-loop*, alcançando alta acurácia global e recall de 100% na detecção de erros em comparação a corretores humanos.

3. Metodologia

Como o objetivo de analisar LLMs na tarefa de avaliação automática de atividades de programação no contexto de disciplinas de POO, este trabalho visa realizar uma avaliação dos principais LLMs atuais na atribuição de notas em atividades práticas e responder a seguinte questão de pesquisa: *"Em que medida LLMs utilizados em suas configurações padrão e sob abordagem zero-shot conseguem produzir avaliações de atividades de Programação Orientada a Objetos alinhadas às avaliações humanas?"*. Para responder a pergunta, este estudo busca mensurar a acurácia e confiabilidade dos principais LLMs atuais, quanto à precisão da nota gerada, quando comparados às notas fornecidas pelo professor da disciplina.

Os modelos foram testados sem alteração de parâmetros de inferência como *temperature* ou *top-p*, utilizando os modelos tal como são disponibilizados em suas respec-

tivas plataformas oficiais. Essa escolha se justifica pois muitos professores que recorrem aos LLMs para gerar correções o fazem por meio de suas plataformas oficiais e não sabem ou não tendem a realizar tais ajustes técnicos nestes parâmetros, o que torna relevante compreender o comportamento e desempenho destes modelos no cenário comum de uso.

Essa escolha também se alinha a estudos recentes, como os de Renze e Guven (2024) e Barros et al. (2025) que apontam que a variação dos parâmetros, especialmente o parâmetro *temperature*, não produz diferenças estatisticamente significativas na qualidade das respostas em tarefas de resolução de programas e geração de *feedback* [Renze and Guven 2024, Barros et al. 2025]. Barros et al. (2025) conclui ainda que estratégias de *prompting*, como *one-shot* e *zero-shot*, são mais determinantes na qualidade das respostas do que os parâmetros supracitados [Barros et al. 2025].

Para realização do experimento, foram selecionadas respostas de uma atividade real, a partir de uma base histórica de atividades da disciplina de Programação Orientada a Objetos. As respostas dos estudantes foram submetidas via ambiente virtual Moodle, em arquivos compactados ou via Github. As respostas foram criteriosamente anonimizadas para a realização do experimento. Ao todo, foram analisadas 46 respostas. Também foi inserido na avaliação o gabarito oficial elaborado pelo professor, totalizando 47 soluções analisadas por cada LLM. O Quadro 1 apresenta descrição da atividade analisada.

No contexto deste trabalho, o foco da atividade foi o conceito de Herança. Essa atividade foi aplicada no primeiro semestre de 2023 nos cursos de Bacharelado em Ciência da Computação e Tecnologia em Sistemas para Internet de uma instituição pública. Os estudantes utilizaram a linguagem de programação Java para realização da atividade proposta. Essa atividade foi selecionada por sua relativa complexidade, onde a resposta correta envolve a combinação de vários arquivos e exige que o estudante aplique diversos conceitos de orientação a objetos. Essa escolha visa avaliar os LLMs em cenários mais desafiadores do que as atividades de disciplinas introdutórias, onde as soluções normalmente são funções únicas e/ou pequenos algoritmos compostos de um único arquivo.

Para fins comparativos, este estudo avalia os LLMs apresentados no Quadro 2. Esses modelos foram selecionados por representarem modelos frequentemente presentes na literatura, além de modelos abertos e fechados mais populares, e por apresentarem desempenho e custo compatíveis para sua utilização em uma futura proposta de ferramenta automatizada de avaliação educacional.

Para a realização do experimento, foi elaborado um *prompt* bem estruturado, seguindo práticas consolidadas de engenharia de *prompt*, incluindo a definição do contexto, os critérios de avaliação, a descrição da atividade e a resposta do estudante. Alinhado aos resultados dos trabalhos de Renze (2024) e Barros et al. (2025), não foram realizados ajustes de parâmetros de inferência como *temperature* ou *top-p*. Ainda considerando o trabalho de Barros et al. (2025), os autores evidenciam um melhor desempenho da abordagem *zero-shot* em problemas mais complexos. Sendo assim, este trabalho adota essa estratégia como abordagem principal alinhando-se às evidências apontadas pela literatura.

Também não foram utilizadas rubricas formais para a correção das atividades. Todos os critérios de avaliação empregados pelo professor durante a correção manual foram incorporados diretamente no *prompt*, organizados no tópico denominado [CRITERIOS]. A ordenação destes critérios seguiu a lógica da importância atribuída pelo docente, indo

Quadro 1: Descrição da atividade proposta aos estudantes

A atividade consiste em, a partir da classe Conta, criada anteriormente e apresentada em outras aulas, criar classes usando os conceitos de Herança. Cada classe herdará de uma superclasse, Conta, que terá todas as informações e comportamentos comuns das classes envolvidas. Cada uma das classes deve ser criada com os seguintes nomes e com as seguintes restrições:

- **Conta Corrente** – Uma conta básica onde o saldo nunca pode ser negativo.
- **Conta Especial** – A conta possui um valor de cheque especial e o saldo da conta pode ser negativo até o valor do cheque especial.
  - **Exemplo**: Saldo = R$1000 e Cheque especial = R$2000, o cliente pode manipular o valor de R$3000. Ao sacar R$3000 o valor do saldo deverá ser - R$2000, nunca ultrapassando o valor do cheque especial;
- **Conta Poupança** – Uma conta onde o saldo não pode ser negativo, mas rende 0,5% a cada um determinado período. No caso do exercício sempre que o método rendimento for executado.
- **Conta Empresarial** – É uma Conta Especial, porém destinada a cliente pessoa jurídica.

Toda conta deve possuir um cliente vinculado. Nenhuma classe Conta deve ter os dados do cliente expressos diretamente na classe. Os dados do cliente devem estar em uma classe específica, classe Cliente, e deve ser vinculado um objeto Cliente com todas as informações do cliente na classe Conta.

Devem ser criados dois tipos de cliente: Pessoa Física e Pessoa Jurídica, sendo que o primeiro terá CPF e o segundo CNPJ. A criação de Pessoa Física e Pessoa Jurídica devem ser feitas usando os conceitos de herança.

Conta Corrente, Conta Poupança e Conta Especial são destinadas às pessoas físicas, enquanto Conta Empresarial é destinada exclusivamente a pessoas jurídicas.

Quadro 2: Modelos de LLMs analisados

Empresa	Modelos
OpenIA	o4-mini, 03-mini, GPT-4.1, GPT-4.1-mini, GPT-4.1-nano, GPT-4o e GPT-4o-mini
Google	Gemini 2.5 pro, Gemini 2.5-flash, Gemini 2.0-flash, Gemini 2.0-flash-lite e Gemini 1.5 pro
DeepSeek	DeepSeek-V3 e DeepSeek-R1
Anthropic	3.7-Sonnet e 3.5-Haiku
X	Grok-3 e Grok-3-mini

do mais relevante para o menos relevante e, consequentemente, do que possuía maior peso de pontuação para o que possuía menor impacto na nota final.

O *prompt* foi enviado igualmente para cada LLM via API (*Application Program-*

ming Interfaces) oficial de cada LLM, utilizando um *script* desenvolvido em Python para automatização desta tarefa. A estrutura do *prompt* é apresentada no Quadro 3. O *script* em Python realiza a descompactação dos arquivos enviados pelos estudantes na plataforma Moodle e extrai os códigos-fonte em Java. É criado um arquivo de texto único por estudante contendo todos os código-fonte em Java submetidos por ele. Em seguida, o arquivo de resposta do estudante é inserido no *prompt* para submissão ao modelo correspondente via API oficial.

Os resultados obtidos por cada LLM foram armazenados em um único arquivo de texto que contém a nota e *feedback* de todos os estudantes avaliados por aquele modelo. Posteriormente, estas notas foram tabuladas para a realização da análise dos dados. A análise quantitativa envolveu a comparação entre as notas atribuídas pelo professor e as notas fornecidas por cada LLM, utilizando as métricas: Kappa Ponderado Quadrático (QWK), Erro Médio Absoluto (*Mean Absolute Error*, MAE), Média da Raiz quadrada do Erro Quadrático (*Root Mean Square Error*, RMSE), Desvio Padrão das Diferenças (*Standard Deviation of the Differences*, SD(D)), Desvio padrão do Erro Absoluto (*Standard Deviation Absolute Error*, SD(AE)) e *p-value*. Adicionalmente, foram calculados os Intervalos de Confiança de 95% para as QWK, MAE e RMSE.

O Kappa Ponderado Quadrático (QWK), uma variação do coeficiente Kappa, atribui diferentes pesos aos desacordos entre avaliadores, penalizando mais fortemente discrepâncias maiores e oferecendo uma análise mais realista da concordância [Cohen 1968].

As métricas MAE e RMSE mensuram o erro entre valores previstos e reais sob diferentes perspectivas. O MAE calcula a média das diferenças absolutas e indica a magnitude do erro, enquanto o RMSE penaliza mais fortemente discrepâncias elevadas por utilizar o quadrado das diferenças, sendo, assim, sensível a *outliers*. Chai e Draxler (2014) recomendam o uso combinado de ambas as métricas para avaliação de modelos, onde cada métrica oferece uma perspectiva diferente sobre os erros dos modelos [Chai and Draxler 2014].

O Desvio Padrão das Diferenças SD(D) quantifica o quanto as notas dos LLMs se afastam das notas do professor, considerando o sinal das diferenças e permitindo identificar vieses de superestimação ou subestimação, enquanto o Desvio Padrão do Erro Absoluto SD(AE) mede a dispersão dos erros absolutos em torno do MAE, indicando o quão homogêneos ou irregulares são esses erros. Valores baixos de SD(D) e SD(AE) sugerem discrepâncias pequenas e consistentes, com alta concordância e desempenho estável dos modelos; já valores altos indicam maior variabilidade, menor alinhamento com a correção humana e maior inconsistência nas avaliações geradas pelos LLMs.

O intervalo de confiança de 95% (IC 95%) indica a faixa de valores dentro da qual se espera que esteja o valor real da métrica na população, considerando a variabilidade dos dados. O *p-value* representa a probabilidade de obtermos um resultado tão extremo quanto o observado (ou mais) assumindo que não exista diferença real entre as notas do modelo e as do professor. Os *p-values* menores ou iguais a 0,05 indicam evidência de diferença estatisticamente significativa, enquanto valores maiores que 0,05 sugerem que não há diferença estatística significativa entres os valores analisados.

Quadro 3: Estrutura do *prompt* utilizado

Você é o professor da disciplina de Programação Orientada a Objetos, de um curso de graduação da área da Computação, e precisa corrigir uma atividade solicitada aos alunos utilizando a linguagem de programação Java, que atenda descrição da atividade, apresentada em [DESCRICAO] e utilizando os critérios apresentados em [CRITERIOS]. Apresente sua resposta usando o [FORMATO_RESPOSTA].

[FORMATO_RESPOSTA]
1. Apresente uma resposta sucinta organizada em 2 partes. (1) Parágrafo curto com comentários gerais sobre a solução enfatizando pontos fortes da solução; (2) Erros cometidos pelo discente na solução.
2. Atribua uma nota para solução apresentada utilizando a escala de 0 a 10 de forma proporcional, onde 10 representa o atendimento integral dos critérios, e reduções devem ser justificadas pelos erros apontados.
3. Não me apresente código de correção da atividade e apenas o que está ou não correto, focando principalmente nos conceitos de Programação Orientada a Objetos, podendo ignorar erros e acertos da main().

[CRITERIOS]
[IMPORTANTE] Os critérios estão pontuados na ordem de importância. Seja mais criterioso em relação aos critérios iniciais.
1. Verifique se todas as regras de negócio definidas na descrição da atividade são atendidas.
2. Atente-se principalmente a possibilidade de atribuir uma pessoa física em uma Conta Empresarial e/ou a possibilidade de atribuir uma Pessoa Jurídica em uma Conta Corrente, Conta Poupança ou Conta Especial, principalmente quando há uma solução que envolva o conceito de Polimorfismo no titular da conta. Caso aconteça essa falha da regra de negócio, apresente o *feedback* e penalize a nota da resposta;
3. A atividade é especialmente elaborada para avaliar a correta aplicação do conceito de Herança, sem Polimorfismo, portanto há a possibilidade do aluno utilizar sobrecarga de métodos e haver uma repetição de código. Isso não deve ser penalizado;
4. A aplicação dos conceitos de Polimorfismo é opcional. Caso o aluno utilize polimorfismo, deve verificar se todas as regras de negócio foram perfeitamente atendidas;
5. Você não deve pontuar os alunos que usaram Polimorfismo ou penalizar os alunos que não usaram este conceito. Foque na utilização do conceito de Herança e na aplicação das regras de negócio estabelecidas na atividade;
6. Caso o aluno tenha utilizado o conceito de polimorfismo de forma incorreta, apresente os erros cometidos no *feedback*;
7. A `main` é opcional portanto pode ignorá-la para atribuição da nota do aluno;
8. Se o discente não enviar uma resposta ou enviar uma resposta vazia, você deve atribuir nota zero e apresentar no *feedback* a falta de envio da resposta. **[DESCRICAO]**

```
//Descrição da atividade apresentado no Quadro 1.
```

[RESPOSTA_ALUNO]
```
//Resposta enviada pelo aluno
```
[RESPOSTA_ALUNO]

4. Análises e discussões

A Tabela 1 apresenta os resultados de cada métrica calculada para cada LLM avaliado. Os valores em negrito destacam os cinco melhores desempenhos em cada métrica. Para as métricas QWK, MAE e RMSE, são exibidos, logo abaixo de cada valor, os respectivos Intervalos de Confiança de 95%, permitindo visualizar a incerteza associada a cada medida.

Tabela 1. Resultado geral das métricas aplicadas

Modelos	QWK	MAE	RMSE	SD(AE)	SD(D)	*p_value*
GPT-4.1	**0,737** [0.536, 0.843]	**1,0319** [0.809, 1.266]	**1,299** [1.054, 1.535]	**0,7968**	**1,3031**	0,751833
Grok-3-mini	**0,728** [0.509, 0.848]	**0,9681** [0.691, 1.266]	**1,393** [1.049, 1.741]	1,0130	**1,399**	0,503184
GPT-4.1-mini	**0,718** [0.504, 0.846]	**1,0319** [0.809, 1.277]	**1,315** [1.024, 1.601]	**0,8236**	**1,3078**	0,285185
Grok-3	**0,711** [0.495, 0.821]	1,1383 [0.883, 1.426]	**1,479** [1.144, 1.819]	**0,9538**	1,4678	0,270046
DeepSeek-V3	**0,708** [0.504, 0.817]	**1,1064** [0.872, 1.351]	**1,391** [1.132, 1.637]	**0,8530**	**1,3794**	0,354095
o4-mini	0,672 [0.499, 0.801]	1,4362 [1.117, 1.777]	1,844 [1.495, 2.177]	1,1685	1,4203	0,000011
o3-mini	0,668 [0.456, 0.801]	1,2234 [0.957, 1.521]	1,563 [1.231, 1.889]	0,9825	1,5018	0,020037
Gemini 2.5-pro	0,664 [0.417, 0.823]	**1,0957** [0.809, 1.404]	1,518 [1.118, 1.899]	1,0614	1,4465	0,041928
Gemini 2.5-flash	0,626 [0.408, 0.767]	1,2872 [0.979, 1.617]	1,715 [1.341, 2.090]	1,1456	1,4934	0,000430
Claude 3.7 Sonnet	0,602 [0.379, 0.755]	1,2553 [0.968, 1.553]	1,608 [1.290, 1.909]	1,0155	**1,401**	0,002460
Gemini 2.0-flash	0,596 [0.359, 0.751]	1,1809 [0.915, 1.457]	1,504 [1.207, 1.779]	**0,9408**	1,425	0,899668
Claude 3.5 Haiku	0,552 [0.319, 0.694]	1,5106 [1.191, 1.851]	1,891 [1.532, 2.218]	1,1491	1,508	0,000027
Gemini 1.5-pro	0,534 [0.250, 0.697]	1,2660 [0.989, 1.553]	1,603 [1.296, 1.905]	0,9937	1,6037	0,671300
DeepSeek-R1	0,510 [0.299, 0.687]	1,5745 [1.202, 1.979]	2,073 [1.619, 2.520]	1,3633	1,5231	0,000001
GPT-4.1-nano	0,426 [0.146, 0.606]	1,4681 [1.149, 1.787]	1,833 [1.493, 2.139]	1,1102	1,6985	0,028894
Gemini 2.0-flash-lite	0,323 [0.122, 0.507]	1,7128 [1.351, 2.117]	2,160 [1.676, 2.638]	1,3300	1,8355	0,000402
GPT-4o	0,276 [0.070, 0.479]	1,8830 [1.436, 2.383]	2,504 [1.845, 3.152]	1,6688	2,1334	0,000169
GPT-4o-mini	0,186 [0.003, 0.383]	1,8723 [1.532, 2.213]	2,224 [1.869, 2.552]	1,2135	1,8648	0,000339

Em uma análise geral, todos os modelos apresentaram valores positivos de QWK, o que indica que há sempre algum grau de concordância com o padrão de correção do

docente. No entanto, a magnitude dessa concordância e o tamanho dos erros variam para cada modelo analisado. Em uma análise conjunta das métricas, é possível separar os LLMs analisados em três grupos.

No grupo de melhor desempenho, composto por GPT-4.1, Grok-3-mini, GPT-4.1-mini, Grok-3 e DeepSeek-V3, os valores de QWK estão entre 0,71 e 0,74. Interpretando estes valores conforme a escala de Landis e Koch (1977) , os modelos apresentam uma concordância "substancial"com o professor, ou seja, os modelos conseguem reproduzir de forma bastante próxima a distribuição das notas do docente, com desvios médios relativamente pequenos. Os modelos apresentaram valores de MAE em torno de 1 ponto na escala de 0 a 10, aliados a RMSE relativamente próximos ao MAE e desvios padrão de erro (SD(AE) e SD(D)) menores, sugerindo que os erros médios são moderados e que grandes discrepâncias são pouco frequentes nesse grupo.

Os intervalos de confiança de 95% para o QWK, MAE e RMSE reforçam o bom desempenho destes modelos. Com uma sobreposição entre os modelos deste grupo, estatisticamente não é possível afirmar que um modelo é superior a outro. O comportamento do *p-value* também é coerente com essa interpretação. Para os cinco modelos deste grupo, os *p-values* calculados são superiores a 0,05 ($p > 0.05$), indicando que não há evidência de diferença estatística significativa entre as avaliações realizadas pelo docente e pelos LLMs deste grupo, sugerindo uma convergência entre a avaliação automatizada e a humana para os critérios analisados.

Um segundo grupo intermediário, inclui modelos como GPT-o4-mini, GPT-o3-mini, Gemini 2.5-pro Gemini 2.5-flash, Claude 3.7 Sonnet, Gemini 2.0-flash, Claude 3.5 Haiku, Gemini 1.5-pro e DeepSeek-R1. Estes modelos apresentaram valores de QWK entre 0,51 e 0,67, MAE entre 1,1 e 1,6 pontos e RMSE na faixa de 1,5 a 2,1. No entanto, a combinação de QWK mais baixo, maior erro médio e maiores dispersões (SD(AE), SD(D)) aponta para uma avaliação menos alinhada com o docente, com discrepâncias mais frequentes e, em vários casos, estatisticamente significativas. Para boa parte desses modelos, os *p-values* ficam abaixo de 0,05 ($p < 0.05$), sugerindo a existência de uma diferença estatisticamente significativa entre as notas atribuídas pelos LLMs e as notas do professor.

Por fim, o terceiro grupo, de menor desempenho, é formado por GPT-4.1-nano, Gemini 2.0-flash-lite, GPT-4o e GPT-4o-mini. Nesses casos, o QWK cai para valores entre 0,19 e 0,43, com erros médios (MAE) entre 1,5 e 1,9 pontos e RMSE acima de 2,0, alcançando até cerca de 2,5. Além disso, tanto SD(AE) quanto SD(D) são mais elevados, indicando maior variabilidade dos erros. Em conjunto com *p-values* sistematicamente inferiores a 0,05 ($p < 0.05$), esses resultados apontam para modelos cuja distribuição de notas se afasta de forma significativa das notas do docente, o que torna seu uso pouco recomendável para qualquer tipo de correção automática sem forte intervenção humana.

A análise conjunta das métricas reforça que essas conclusões não dependem de um único indicador. Há uma relação consistente em que maiores valores de QWK associam-se a menores MAE e RMSE e a menor variabilidade dos erros (SD(AE) e SD(D)). Essa coerência interna sugere que um bom grau de alinhamento das métrica selecionadas e confirma que apenas um subconjunto de LLMs atinge um patamar de concordância próximo ao docente em modo *zero-shot*.

Um ponto importante é a equivalência, e em certos casos superioridade, dos modelos de menor custo computacional (mini) frente aos modelos tradicionais ou *flagships*. O modelo Grok-3-mini, superou sua versão maior (Grok-3) e destacou-se por apresentar o menor MAE de todo o experimento (MAE = 0,968), sendo o único a apresentar valor abaixo 1,0 ponto de erro médio. Similarmente, o GPT-4.1-mini (QWK=0,718) apresentou desempenho estatisticamente indistinguível de sua versão maior, GPT-4.1 (QWK=0,737), consolidando-se como uma alternativa de menor custo e alta eficiência para implementação em larga escala.

Em contraste, modelos de *reasoning*, focados em cadeia de pensamento (*Chain-of-Thought*), como é o caso do GPT-o4-mini e GPT-o3-mini, apresentaram um comportamento inferior em relação aos modelos padrões. Embora tenham alcançado bons índices de concordância (QWK de 0,672, 0,668, respectivamente), posicionando-se próximo do melhor grupo do experimento, ambos apresentaram diferença estatística significativa ($p < 0,05$) entre as notas atribuídas pelos modelos e o docente. Já o DeepSeek-R1, obteve um QWK de apenas 0.510 e um MAE elevado de 1.57. Os *p-values* extremamente baixos para o o4-mini ($p < 0.001$) e DeepSeek-R1 ($p < 0,000001$) confirmam que suas distribuições de notas desviam significativamente do padrão humano.

Por fim, nota-se a obsolescência rápida de modelos que eram referência até pouco tempo. O GPT-4o e GPT-4o-mini apresentaram os piores desempenhos do estudo (QWK de 0.276 e 0.186, respectivamente), indicando que, para tarefas de avaliação sensível ao contexto acadêmico, as versões mais recentes são mais indicadas.

Em síntese, os resultados indicam que LLMs em configuração padrão e sob abordagem *zero-shot* podem produzir avaliações de atividades de POO alinhadas às humanas. Para os modelos do grupo de melhor desempenho, os indicadores sugerem a viabilidade de uso em correções semi-automatizadas, especialmente em abordagens *human-in-the-loop* nas quais o professor mantém o controle e valida as decisões. Já os modelos de desempenho intermediário e baixo demandam maior cautela, ajustes de *prompt*, calibração adicional ou restrição de uso a tarefas de apoio menos sensíveis à precisão da nota, reforçando que a adoção de LLMs na avaliação deve ser feita de forma criteriosa, mas exige seleção cuidadosa do modelo e desenho cuidadoso do contexto de uso.

4.1. Limitações da pesquisa

O presente estudo apresenta contribuições para o uso de LLMs na avaliação de atividades de programação aplicadas ao contexto de POO, mas alguns fatores podem restringir a generalização dos resultados. Nenhuma rubrica formal de correção foi fornecida aos LLMs. Cada modelo, julgou as soluções segundo os critérios informados no *prompt* mas sem a devida definição de uma rubrica para alinhar a avaliação. A falta desta rubrica pode não garantir que a nota reflita os objetivos de aprendizagem nem que avaliações diferentes sejam comparáveis entre si.

A decisão de não alterar parâmetros de inferência, como *temperature* e *top-p*, e o uso da estratégia de *prompting zero-shot* foram baseadas em trabalhos aplicados ao contexto de programação introdutória. Aplicados ao contexto de programação orientada a objetos, a variação destas configurações podem gerar resultados diferentes dos relatados nos trabalhos Renze e Guven (2024) e Barros et al. (2025) .

Além disso, o experimento utilizou apenas uma tarefa de Programação Orientada

a Objetos (Herança) escrita em uma única linguagem de programação (Java), avaliando somente 47 submissões, sendo 46 respostas de alunos e o gabarito do professor. Esse escopo limitado reduz a validade e impede concluir se o desempenho se mantém em domínios ou níveis de dificuldade distintos.

5. Considerações Finais

Este estudo teve como objetivo avaliar a precisão e acurácia dos principais LLMs atuais aplicados na correção automática de uma atividade prática de programação da disciplina de POO. Os resultados obtidos permitem responder de forma direta à pergunta de pesquisa sobre em que medida LLMs, em configuração padrão e sob abordagem *zero-shot*, conseguem produzir avaliações de atividades de Programação Orientada a Objetos alinhadas às avaliações humanas.

Para isso, foi conduzida uma análise quantitativa baseada em um conjunto de métricas estatísticas complementares, incluindo o Kappa quadrático ponderado (QWK), o Erro Médio Absoluto (MAE), a Raiz do Erro Quadrático Médio (RMSE), o Desvio Padrão das Diferenças (SD(D)) e o Desvio Padrão do Erro Absoluto (SD(AE)), além do cálculo de intervalos de confiança de 95% para QWK, MAE e RMSE e calculo do *p-value* com a aplicação do teste de Wilcoxon pareado entre as notas dos modelos e as notas atribuídas pelo docente. Essa combinação de métricas permitiu não apenas quantificar o erro médio dos modelos, mas também avaliar sua estabilidade, o grau de concordância com o professor e a significância estatística das diferenças observadas.

Os resultados demonstram a formação de um grupo de modelos com melhor desempenho, composto por GPT-4.1, GPT-4.1-mini, Grok-3-mini, Grok-3 e DeepSeek-V3, que apresentaram valores de QWK na faixa de concordância substancial em relação ao docente e MAE em torno de 1 ponto em uma escala de 0 a 10, com RMSE e desvios padrão relativamente baixos, indicando baixa frequência de discrepâncias extremas. A análise dos intervalos de confiança de 95% para QWK, MAE e RMSE mostra forte sobreposição entre os modelos deste grupo, configurando um "empate técnico" em termos de concordância e erro médio, o que sugere que esses modelos replicam a avaliação do professor com precisão semelhante.

Adicionalmente, os testes de Wilcoxon indicaram *p-values* superiores a 0,05 ($p > 0.05$) para esse conjunto de modelos, não havendo evidência estatística de diferença sistemática entre as distribuições de notas desses LLMs e as notas humanas. Sendo assim, dentro do escopo deste experimento, tais modelos podem ser considerados estatisticamente equivalentes ao docente no que se refere à distribuição das notas atribuídas.

Como trabalhos futuros, pretende-se ampliar o conjunto de atividades avaliadas, contemplar diferentes tópicos de POO e múltiplas linguagens de programação e realizar a análise qualitativa dos *feedbacks* gerados pelos modelos, investigando seu alinhamento pedagógico, clareza e utilidade para os estudantes. Além disso, realizar estudos para explorar o impacto da variação de parâmetros de inferência, como *temperature* e *top-p*, na qualidade da correção e na ocorrência de alucinações, avaliando o uso desses modelos em cenários reais de sala de aula, considerando a percepção de docentes e discentes sobre a confiabilidade das notas e dos *feedbacks* produzidos. Esses desdobramentos são fundamentais para consolidar o uso de LLMs como ferramentas de apoio à correção e ao *feedback* em larga escala, de forma seguram, transparente e pedagogicamente alinhada.

Agradecimentos

Os autores agradecem o apoio do Instituto Federal de Educação, Ciência e Tecnologia Goiano - IF Goiano e do Programa de Pós-graduação da Faculdade de Computação da Universidade Federal de Uberlândia (PPGCO/FACOM/UFU).

Uso de Inteligência Artificial

Os autores declaram que ferramentas de inteligência artificial, como Modelos de Grande Escala (LLMs), não foram empregados na redação ou revisão deste texto.

Referências

Akyash, M., Azar, K. Z., and Kamali, H. M. (2025). StepGrade: Grading Programming Assignments with Context-Aware LLMs. arXiv:2503.20851 [cs].

Balse, R., Kumar, V., Prasad, P., and Warriem, J. M. (2023). Evaluating the Quality of LLM-Generated Explanations for Logical Errors in CS1 Student Programs. In *Proceedings of the 16th Annual ACM India Compute Conference*, COMPUTE '23, pages 49–54, New York, NY, USA. Association for Computing Machinery.

Barros, J., Moraes, L. O., Oliveira, F., and Delgado, C. A. D. M. (2025). Large Language Models Generating Feedback for Students of Introductory Programming Courses. In Cristea, A. I., Walker, E., Lu, Y., Santos, O. C., and Isotani, S., editors, *Artificial Intelligence in Education*, pages 421–433, Cham. Springer Nature Switzerland.

Chai, T. and Draxler, R. R. (2014). Root mean square error (rmse) or mean absolute error (mae)? – arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250.

Cohen, J. (1968). Weighted kappa: nominal scale agreement with provision for scaled disagreement or partial credit. *Psychological Bulletin*, 70(4):213–220.

Efan, E., Krismadinata, K., Jama, J., and Mulya, R. (2023). A Systematic Literature Review of Teaching and Learning on Object-Oriented Programming Course. *International Journal of Information and Education Technology*, 13:302–312.

Estévez-Ayres, I., Callejo, P., Hombrados-Herrera, M. A., Alario-Hoyos, C., and Delgado Kloos, C. (2025). Evaluation of LLM Tools for Feedback Generation in a Course on Concurrent Programming. *International Journal of Artificial Intelligence in Education*, 35(2):774–790.

Ferreira Mello, R., Pereira Junior, C., Rodrigues, L., Pereira, F. D., Cabral, L., Costa, N., Ramalho, G., and Gasevic, D. (2025). Automatic short answer grading in the llm era: Does gpt-4 with prompt engineering beat traditional models? In *Proceedings of the 15th international learning analytics and knowledge conference*, pages 93–103.

Grandel, S., Schmidt, D. C., and Leach, K. (2024). Applying Large Language Models to Enhance the Assessment of Parallel Functional Programming Assignments. In *Proceedings of the 1st International Workshop on Large Language Models for Code*, LLM4Code '24, pages 102–110, New York, NY, USA. Association for Computing Machinery.

Gutiérrez, L. E., Guerrero, C. A., and López-Ospina, H. A. (2022). Ranking of problems and solutions in the teaching and learning of object-oriented programming. *Education and Information Technologies*, 27(5):7205–7239.

Kasneci, E., Sessler, K., Küchemann, S., Bannert, M., Dementieva, D., Fischer, F., Gasser, U., Groh, G., Günnemann, S., Hüllermeier, E., Krusche, S., Kutyniok, G., Michaeli, T., Nerdel, C., Pfeffer, J., Poquet, O., Sailer, M., Schmidt, A., Seidel, T., Stadler, M., Weller, J., Kuhn, J., and Kasneci, G. (2023). ChatGPT for good? On opportunities and challenges of large language models for education. *Learning and Individual Differences*, 103:102274.

Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.

Lobo, J., Anthony, L., Falcao, A., Xavier, C., Torrezão, N., Isotani, S., Ibert, I., Rodrigues, L., and Mello, R. (2025). Automatic scoring of elementary school essays in brazilian portuguese with llms: Comparing gemini, gpt-4o, claude, and mistral. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 167–180. SBC.

Marques, D. and Morandini, M. (2024). Uso do ChatGPT no Contexto Educacional: Uma Revisão Sistemática da Literatura. In *Simpósio Brasileiro de Informática na Educação (SBIE)*, pages 1784–1795. SBC. ISSN: 0000-0000.

Martins, R. M. (2025). Java-Judge-OO: Uma Ferramenta Educacional para Avaliação Automatizada de Programação Orientada a Objetos em Java. In *Simpósio Brasileiro de Educação em Computação (EDUCOMP)*, pages 39–41. SBC. ISSN: 3086-0741.

Menolli, A. and Strik, B. (2025). Educational Insights from Code: Mapping Learning Challenges in Object-Oriented Programming through Code-Based Evidence. In *Simpósio Brasileiro de Engenharia de Software (SBES)*, pages 544–554. SBC. ISSN: 2833-0633.

Montenegro-Rueda, M., Fernández-Cerero, J., Fernández-Batanero, J. M., and López-Meneses, E. (2023). Impact of the Implementation of ChatGPT in Education: A Systematic Review. *Computers*, 12(8):153. Number: 8 Publisher: Multidisciplinary Digital Publishing Institute.

Nakamoto, R., Flanagan, B., Yamauchi, T., Dai, Y., Takami, K., and Ogata, H. (2023). Enhancing automated scoring of math self-explanation quality using llm-generated datasets: A semi-supervised approach. *Computers*, 12(11):217.

Quincozes, C., Molinos, D., Araújo, R., Quincozes, S., and Guedes, G. (2025). Engenharia de prompt para a geração automatizada de questões assistida por LLMs: Uma análise comparativa. In *Anais do XXXVI Simpósio Brasileiro de Informática na Educação*, pages 1347–1360, Porto Alegre, RS, Brasil. SBC.

Razafinirina, M. A., Dimbisoa, W. G., and Mahatody, T. (2024). Pedagogical Alignment of Large Language Models (LLM) for Personalized Learning: A Survey, Trends and Challenges. *Journal of Intelligent Learning Systems and Applications*, 16(4):448–480. Number: 4 Publisher: Scientific Research Publishing.

Renze, M. and Guven, E. (2024). The Effect of Sampling Temperature on Problem Solving in Large Language Models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 7346–7356. arXiv:2402.05201 [cs].

Seo, H., Hwang, T., Jung, J., Kang, H., Namgoong, H., Lee, Y., and Jung, S. (2025). Large Language Models as Evaluators in Education: Verification of Feedback Consistency and Accuracy. *Applied Sciences*, 15(2):671. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

Silva, P. and Costa, E. (2025). Assessing Large Language Models for Automated Feedback Generation in Learning Programming Problem Solving. arXiv:2503.14630 [cs].

Strik, B. H. (2025). Uma abordagem baseada em inteligência artificial para identificação e classificação automatizada de problemas na aprendizagem de programação orientada a objetos por meio da análise de código-fonte. Master's thesis, State University of Londrina, Londrina.

Ta, N. B. D., Nguyen, H. G. P., and Gottipati, S. (2023). ExGen: Ready-To-Use Exercise Generation in Introductory Programming Courses. *International Conference on Computers in Education*.

Xie, W., Niu, J., Xue, C. J., and Guan, N. (2024). Grade Like a Human: Rethinking Automated Assessment with Large Language Models. arXiv:2405.19694 [cs].

Yousef, M., Mohamed, K., Medhat, W., Mohamed, E. H., Khoriba, G., and Arafa, T. (2025). BeGrading: large language models for enhanced feedback in programming education. *Neural Computing and Applications*, 37(2):1027–1040.