Avaliação Sistemática de Estratégias de Correção Automática de Questões Abertas em Engenharia de Software com LLMs

Projeto Orientado em Computação II

Iago Gabino Gonçalves

Orientador: Marco Túlio de Oliveira Valente

[1] Departamento de Ciência da Computação – UFMG
Belo Horizonte – MG – Brasil

Abstract. *This report presents the second stage of a two-stage undergraduate research project on automatic grading of open-ended Software Engineering questions with large language models. While the first stage established the technical feasibility of the platform, the second stage focused on systematically comparing grading protocols under a fixed model. The work produced a reorganized question bank, a semi-automatic question creation and review pipeline, a controlled set of reference answers, and an experimental evaluation of three grading protocols: direct grading, grading with review, and aggregated grading. In the analyzed experimental setting, aggregated grading achieved the lowest rubric-level error, direct grading offered the best cost-benefit trade-off, and the official review protocol did not improve the baseline. Additional sensitivity analyses suggest that better calibrated review prompts are a promising direction for future validation.*

Resumo. *Este relatório apresenta a segunda etapa de um projeto sobre correção automática de questões abertas de Engenharia de Software com modelos de linguagem. Enquanto o Projeto Orientado em Computação I (POC I) estabeleceu a viabilidade técnica da plataforma, o Projeto Orientado em Computação II (POC II) concentrou-se em comparar sistematicamente protocolos de correção usando um modelo fixo. O trabalho produziu um banco de questões reorganizado, um pipeline semiautomático de criação e revisão, um conjunto controlado de respostas de referência e uma avaliação experimental de três protocolos: avaliação direta, avaliação com revisão e avaliações agregadas. No recorte experimental analisado, as avaliações agregadas obtiveram o menor erro em nível de rubrica, a avaliação direta apresentou o melhor custo-benefício nesse recorte e a revisão oficial não melhorou a linha de base. Análises adicionais indicam que prompts de revisão melhor calibrados constituem uma direção promissora para validação futura.*

1. Introdução

Questões abertas são uma forma importante de avaliar competências em Engenharia de Software. Em muitos cenários, o que se deseja medir não é apenas se um programa executa corretamente, mas se o estudante identifica a causa de um defeito, preserva contratos de uma refatoração, propõe testes significativos, justifica decisões arquiteturais ou reconhece trade-offs entre alternativas técnicas. Esses aspectos são difíceis de reduzir a

testes automatizados tradicionais, pois envolvem argumentação, interpretação de contexto e avaliação por critérios.

A correção manual desse tipo de resposta, entretanto, tem custo elevado. Além do tempo necessário para ler respostas textuais ou mistas com código, há variação entre avaliadores e dificuldade de manter consistência quando a escala aumenta. Modelos de linguagem de grande porte oferecem uma alternativa promissora: eles podem ler enunciado, resposta e rubrica, produzindo uma avaliação estruturada. Porém, usar um modelo como avaliador não resolve automaticamente o problema. A qualidade do julgamento depende do formato da questão, da clareza da rubrica, do prompt de avaliação, da validação da saída e do protocolo usado para chamar o modelo.

O Projeto Orientado em Computação I estabeleceu a viabilidade técnica inicial de uma plataforma para correção automática de questões abertas de Engenharia de Software com apoio de modelos de linguagem. O sistema já era capaz de carregar questões estruturadas, montar prompts a partir de rubricas, chamar um modelo, validar a saída em JSON e registrar avaliações. O Projeto Orientado em Computação II avançou sobre essa base com uma pergunta mais específica:

> *Qual protocolo de correção com LLM produz avaliações mais aderentes à rubrica em questões abertas de Engenharia de Software?*

Neste trabalho, a comparação não foi feita entre modelos diferentes. Todas as estratégias oficiais usaram o mesmo modelo, Gemini 2.5 Flash via OpenRouter. Essa decisão metodológica foi central: manter o modelo fixo permite isolar o efeito do protocolo de avaliação. Se modelos diferentes fossem comparados ao mesmo tempo, não seria possível distinguir se uma diferença de desempenho veio da capacidade do modelo ou da forma de conduzir a correção.

As principais contribuições do POC II foram:

1. reorganização e expansão do banco de questões por eixo, subtema, dificuldade e tipo de resposta;
2. definição de um pipeline semiautomático de criação e revisão de questões, com uso assistivo de LLMs e curadoria humana;
3. construção de um conjunto controlado de respostas de referência, com notas esperadas por critério;
4. implementação e comparação de três protocolos de correção automática;
5. análise quantitativa de aderência à rubrica, custo, latência e robustez;
6. análise exploratória de versões alternativas do prompt de revisão.

O restante do relatório está organizado da seguinte forma. A Seção 2 apresenta o referencial teórico. A Seção 3 resume a contribuição do POC II. As Seções 4, 5 e 6 descrevem o banco de questões, o pipeline de criação e as respostas de referência. As Seções 7, 8, 9, 10 e 11 apresentam o desenho experimental, as estratégias, as métricas, os resultados e os diagnósticos. Por fim, as Seções 12, 13, 14 e 15 discutem achados, limitações, conclusão e trabalhos futuros.

2. Referencial Teórico e Trabalhos Relacionados

Este trabalho se apoia em três linhas principais: correção automática de respostas abertas, uso de modelos de linguagem como avaliadores e protocolos com múltiplos julgamentos.

2.1. Correção automática de respostas abertas

A área de *Automatic Short Answer Grading* investiga métodos para atribuir notas a respostas textuais curtas ou médias. Burrows et al. [1] mostram que o problema vai além de similaridade lexical: uma resposta pode usar palavras diferentes da referência e ainda estar conceitualmente correta. Em Engenharia de Software, essa dificuldade é ampliada porque uma resposta pode misturar código, justificativa, diagnóstico, trade-offs e decisões de projeto.

Trabalhos recentes investigam modelos de linguagem para pontuação de respostas abertas. Chamieh et al. [7] discutem limitações e oportunidades de abordagens sem exemplos e com poucos exemplos. Xie et al. [8] defendem que a avaliação automatizada deve ser tratada como um processo completo, envolvendo rubrica, julgamento e revisão. Essa perspectiva é compatível com o presente trabalho: a contribuição não é apenas chamar um modelo, mas comparar protocolos de correção sob a mesma base experimental.

2.2. Modelos de linguagem como avaliadores

O paradigma de usar LLMs como avaliadores ganhou força em tarefas nas quais métricas automáticas tradicionais são insuficientes. Zheng et al. [3] mostram que modelos fortes podem se aproximar de preferências humanas em avaliações abertas, mas também destacam vieses como preferência por respostas mais longas ou por determinadas posições. Chiang e Lee [2] analisam modelos de linguagem como alternativa à avaliação humana e reforçam que o desenho experimental influencia fortemente os resultados.

Liu et al. [4] propõem o G-Eval, no qual critérios explícitos e saída estruturada ajudam a aproximar o julgamento automático de avaliações humanas. Para este projeto, a implicação é direta: o modelo não deve atuar como avaliador genérico. Ele precisa receber enunciado, rubrica, solução esperada e resposta avaliada, além de retornar um formato estruturado que possa ser validado.

Em contexto educacional, Chiang et al. [9] relatam o uso de LLMs como avaliadores em uma disciplina com mais de mil estudantes, destacando tanto a utilidade prática da abordagem quanto desafios de aderência às instruções e confiabilidade.

2.3. Rubricas analíticas e avaliação por critério

O uso de rubricas analíticas é um elemento metodológico central deste trabalho. Em vez de atribuir apenas uma nota global, a resposta é decomposta em critérios observáveis, cada um com peso e descrição própria. Essa estrutura melhora a auditabilidade da correção: torna explícito se a nota veio de correção funcional, justificativa técnica, cobertura de casos de borda, preservação de contrato, análise de trade-offs ou outra dimensão da competência avaliada.

Essa escolha também reduz ambiguidades na avaliação automática. Quando o modelo recebe uma rubrica explícita, ele deixa de operar apenas por impressão geral da resposta e passa a comparar evidências contra critérios definidos. Além disso, a avaliação por critério permite medir erros que a nota global esconderia. Duas avaliações podem ter a mesma nota final, mas uma delas pode distribuir os pontos de forma desalinhada com a rubrica. Por isso, o erro por critério é tratado neste relatório como métrica fundamental de aderência.

2.4. Múltiplos julgamentos e agregação

Protocolos com múltiplas chamadas ao modelo podem reduzir variabilidade e explorar perspectivas complementares. Wang et al. [5] mostram que amostragens múltiplas e agregação podem melhorar tarefas de raciocínio. Por outro lado, Wang et al. [6] mostram que avaliadores baseados em LLMs podem apresentar vieses sistemáticos. Portanto, múltiplas chamadas não eliminam automaticamente o erro: se todas compartilham o mesmo viés, a agregação apenas estabiliza um julgamento incorreto.

Este trabalho adapta essa ideia para correção por rubrica. Em vez de usar modelos diferentes, as avaliações agregadas usam o mesmo modelo em três avaliações independentes, cada uma com foco instrucional distinto: aderência estrita à rubrica, evidências textuais e cobertura conceitual.

3. Visão Geral da Contribuição

O POC II consolidou uma mudança de foco em relação ao POC I. A primeira etapa demonstrou que a plataforma de correção automática era tecnicamente viável. A segunda etapa transformou a plataforma em instrumento experimental: o objetivo passou a ser comparar estratégias de correção, medir seus erros e discutir seus custos.

A contribuição do projeto não é afirmar que LLMs corrigem respostas abertas de forma universal. A contribuição é mostrar, em um ambiente controlado, que o protocolo de avaliação altera os resultados mesmo quando o modelo permanece o mesmo. Essa distinção é importante em aplicações educacionais: uma solução mais cara, com mais chamadas ao modelo, só é justificável se trouxer ganho mensurável em aderência à rubrica ou em confiabilidade.

O projeto também consolidou o banco de questões como artefato experimental. As questões não foram tratadas apenas como conteúdo para uma interface. Elas passaram a ser organizadas por taxonomia, rubrica, tipo de resposta, solução esperada, referências sugeridas e revisão de qualidade. Essa organização permite tanto o uso educacional quanto a seleção controlada de amostras experimentais.

4. Banco de Questões

O banco de questões final foi organizado em quatro eixos principais: correção de bugs, qualidade de código, testes e arquitetura. Cada eixo representa uma família de competências práticas em Engenharia de Software. Dentro de cada eixo, subtemas refinam o tipo de habilidade avaliada e ajudam a manter diversidade entre os problemas propostos.

A Tabela 1 apresenta a taxonomia usada no banco final.

Tabela 1. Eixos e subtemas do banco de questões

Eixo	Subtemas
Correção de bugs	Assincronismo, estado e ciclo de vida, erros em tempo de execução, integração e desempenho.
Qualidade de código	Duplicação e reúso, projeto e refatoração, modularidade, legibilidade.
Testes	Testes unitários e de integração, testabilidade, projeto de testes, cobertura e qualidade.
Arquitetura	Desenho de APIs, desenho de sistemas, escalabilidade e desempenho, decisões arquiteturais.

O banco final do POC II conta com 57 questões. A distribuição não é perfeitamente uniforme entre eixos, mas mantém cobertura ampla das quatro famílias de competência. A Tabela 2 mostra a distribuição por eixo e a Tabela 3 mostra a distribuição por dificuldade.

Tabela 2. Distribuição por eixo

Eixo	Questões
Correção de bugs	14
Qualidade de código	14
Testes	15
Arquitetura	14
Total	57

Tabela 3. Distribuição por dificuldade

Dificuldade	Questões
Fácil	17
Intermediária	24
Avançada	16

Tabela 4. Distribuição cruzada por eixo e dificuldade

Eixo	Fácil	Intermediária	Avançada	Total
Correção de bugs	4	6	4	14
Qualidade de código	4	6	4	14
Testes	5	6	4	15
Arquitetura	4	6	4	14

A distribuição por dificuldade reflete uma escolha de projeto. Questões fáceis cobrem fundamentos e ajudam a validar se a estratégia de correção reconhece respostas

simples e bem delimitadas. Questões intermediárias concentram a maior parte do banco porque exigem raciocínio técnico relevante sem transformar cada item em um estudo de caso extenso. Questões avançadas, em proporção menor que as intermediárias, ampliam o banco para cenários com concorrência, resiliência, decisões arquiteturais com custos e benefícios, e abstrações mais sofisticadas.

Além da dificuldade, o tipo de resposta também foi controlado. No banco final, 43 questões exigem resposta com código e 14 exigem resposta textual organizada em seções. Essa distinção é relevante porque as estratégias de correção precisam lidar tanto com artefatos implementáveis quanto com raciocínio arquitetural.

Tabela 5. Distribuição por eixo e subtema

Eixo	Subtema	Questões
Correção de bugs	Integração e desempenho	4
Correção de bugs	Assincronismo	3
Correção de bugs	Estado e ciclo de vida	3
Correção de bugs	Erros em tempo de execução	4
Qualidade de código	Modularidade	4
Qualidade de código	Projeto e refatoração	4
Qualidade de código	Duplicação e reúso	3
Qualidade de código	Legibilidade	3
Testes	Cobertura e qualidade	4
Testes	Testabilidade	4
Testes	Projeto de testes	4
Testes	Testes unitários e de integração	3
Arquitetura	Decisões arquiteturais	3
Arquitetura	Desenho de APIs	3
Arquitetura	Desenho de sistemas	4
Arquitetura	Escalabilidade e desempenho	4

5. Pipeline Semiautomático de Criação e Revisão

A criação das questões seguiu um pipeline semiautomático. O uso de LLMs foi assistivo, não autônomo: o modelo ajudou a gerar e revisar material, mas a decisão final permaneceu humana. No pipeline usado neste trabalho, o agente gerador foi executado com Claude Sonnet 4.6 Thinking via Antigravity, enquanto o agente revisor foi executado com GPT-5.5 High via Codex. A Figura 1 resume o fluxo.

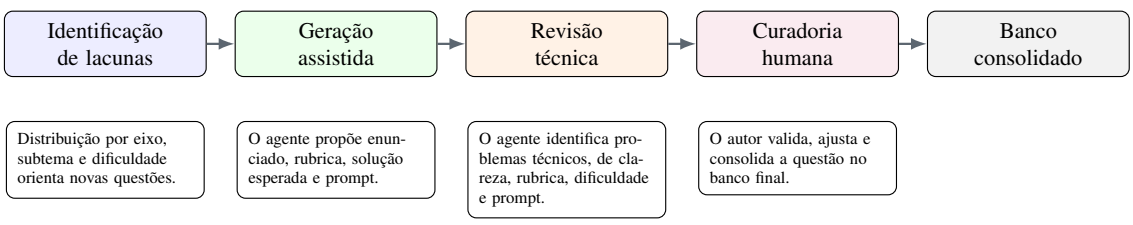

Figura 1. Pipeline de criação e revisão das questões do banco

O primeiro passo do pipeline foi identificar lacunas de cobertura. Em vez de gerar questões isoladas, o processo partiu da distribuição desejada por eixo, subtema e dificuldade. Isso evitou redundância e permitiu direcionar a composição final do banco para uma cobertura equilibrada entre eixos, subtemas e dificuldades.

O segundo passo foi a geração assistida. O agente gerador foi instruído a criar questões práticas, realistas e avaliáveis por rubrica, sempre respeitando a taxonomia oficial do banco. Ele não deveria inventar novos eixos, subtemas ou formatos. Antes de produzir uma questão, a instrução exigia consultar padrões já existentes de questões e prompts, escolher exemplos do mesmo eixo e, quando possível, do mesmo tipo de resposta. O resultado esperado era sempre um par de artefatos: a questão e o prompt de avaliação correspondente.

A instrução do gerador também definia o formato estrutural das questões. Questões de código, usadas em correção de bugs, qualidade de código e testes, deveriam conter frontmatter YAML, metadados, rubrica, enunciado, templates por linguagem e solução esperada. Questões de arquitetura deveriam usar resposta textual em seções, sem templates de linguagem, com enunciado, requisitos, restrições e solução esperada orientada por trade-offs. Em ambos os casos, o agente deveria registrar referências sugeridas, métricas mínimas de geração, critérios observáveis e pesos que somassem 1,0. O prompt de avaliação gerado junto com a questão deveria ser específico para aquela rubrica, usar os placeholders compatíveis com o tipo de resposta e exigir saída estruturada em JSON com nota global, critérios e feedback.

O terceiro passo foi a revisão técnica. O agente revisor foi instruído a auditar a questão candidata e seu prompt de avaliação, mas não a reescrevê-los. Essa separação foi proposital: o material gerado, os problemas identificados e a decisão humana final deveriam permanecer distinguíveis. O revisor avaliava frontmatter, coerência entre identificadores, eixo, subtema, dificuldade, tipo de resposta, linguagens suportadas, enunciado, templates, seções de resposta, rubrica, pesos, solução esperada, prompt de avaliação, placeholders e referências.

A etapa final foi a curadoria humana. Nessa etapa, as recomendações do agente revisor foram analisadas manualmente pelo autor e incorporadas quando pertinentes. A curadoria envolveu ajustes de clareza no enunciado, refinamento de rubricas, correção de inconsistências técnicas, adequação de dificuldade, revisão de templates e validação do prompt de avaliação. Nenhuma questão candidata precisou ser rejeitada integralmente: os problemas identificados foram resolvidos por ajustes leves ou moderados durante a consolidação. Esse resultado sugere que modelos de maior capacidade, quando guiados por uma taxonomia explícita, exemplos existentes, formato de saída bem definido e critérios claros de qualidade, conseguem produzir questões já próximas do padrão esperado. Ainda assim, a curadoria humana permaneceu necessária para garantir consistência técnica, alinhamento pedagógico e integração final das questões ao banco do POC II.

A revisão avaliava cinco dimensões: correção técnica, clareza do enunciado, alinhamento entre enunciado e rubrica, adequação da dificuldade e alinhamento do prompt de avaliação. Além das notas, o revisor indicava uma recomendação de aceite, revisão leve, revisão maior ou rejeição; classificava o principal tipo de problema; apontava as partes que receberam atenção na curadoria; e registrava comentários objetivos para orien-

tar a consolidação final das questões. A Tabela 6 reescreve essas dimensões em termos interpretáveis para o leitor.

Tabela 6. Dimensões usadas pelo agente revisor de questões

Dimensão	Interpretação	Média
Correção técnica	Verifica se o problema, o código, a solução esperada e os conceitos usados são tecnicamente consistentes.	4,59
Clareza do enunciado	Avalia se a tarefa é compreensível, delimitada e suficiente para orientar a resposta do estudante.	4,88
Alinhamento enunciado-rubrica	Mede se a rubrica realmente avalia o que o enunciado pede e se os critérios são observáveis.	4,76
Adequação da dificuldade	Verifica se a dificuldade declarada corresponde ao raciocínio exigido.	4,98
Alinhamento do prompt	Avalia se o prompt de correção está compatível com a rubrica e com o tipo de resposta esperado.	4,76

Essas médias foram calculadas sobre as 41 questões do banco final que passaram por revisão estruturada. Em todas elas, o formato esperado foi considerado válido pelo agente revisor. A Tabela 7 mostra a distribuição das recomendações.

Tabela 7. Recomendações do agente revisor no subconjunto revisado

Recomendação	Questões
Aceitar sem mudança material	23
Revisão leve por curadoria humana	17
Revisão moderada	1

Tabela 8. Principais achados do agente revisor

Classificação principal	Questões
Sem problema material	23
Correção técnica	10
Prompt de avaliação	3
Clareza	2
Rubrica	2
Formato	1

O resultado não deve ser lido como uma substituição da curadoria humana, mas como evidência de que o processo assistido por LLMs produziu candidatos tecnicamente

aproveitáveis e compatíveis com o formato esperado. O papel do revisor foi apontar riscos e orientar a revisão humana. Nos casos com problemas materiais, os achados se concentraram principalmente em correção técnica, prompt de avaliação, clareza, rubrica e formato. As partes mais frequentemente sugeridas para alteração foram prompt, solução esperada, rubrica, enunciado e templates, com uma ocorrência adicional relacionada a referências. Assim, o pipeline funcionou como um mecanismo de triagem e consolidação: ele tornou explícitos os pontos revisados antes da incorporação das questões ao banco final e da seleção da amostra experimental.

6. Respostas de Referência

As respostas de referência foram um artefato central do experimento. Elas permitiram comparar as estratégias de correção contra uma expectativa definida previamente por curadoria humana. Cada resposta simulava uma possível submissão de estudante e era acompanhada de notas esperadas por critério da rubrica, definidas manualmente a partir da solução esperada da questão.

O objetivo não foi criar respostas perfeitas nem reproduzir respostas reais de estudantes. O objetivo foi construir um conjunto controlado, no qual cada questão tivesse três níveis de qualidade:

- **Forte**: resposta correta, completa e bem alinhada à rubrica;
- **Intermediária**: resposta parcialmente correta, com lacunas relevantes;
- **Fraca ou enganosa**: resposta incompleta, incorreta ou superficialmente plausível, mas pouco aderente à rubrica.

A elaboração dessas respostas foi uma etapa manual e iterativa do experimento. Para cada questão selecionada, foram analisados o enunciado, a rubrica, os pesos dos critérios e a solução esperada. A partir desse material, foi identificado o núcleo da competência avaliada: por exemplo, corrigir um erro de execução, preservar comportamento em uma refatoração, propor testes relevantes ou justificar uma decisão arquitetural. Em seguida, foram escritas três respostas com variação controlada de qualidade, evitando uma separação superficial entre respostas “certas”, “parcialmente certas” e “erradas”. Cada resposta precisava ser plausível como submissão de estudante, suficientemente distinta das demais e avaliável pelos mesmos critérios da rubrica. Além disso, as notas esperadas foram atribuídas manualmente por critério, com justificativas associadas, para que o experimento pudesse comparar as estratégias de correção contra uma referência explícita e não apenas contra uma impressão geral do autor.

A Figura 2 resume a relação entre questão, rubrica e respostas de referência.

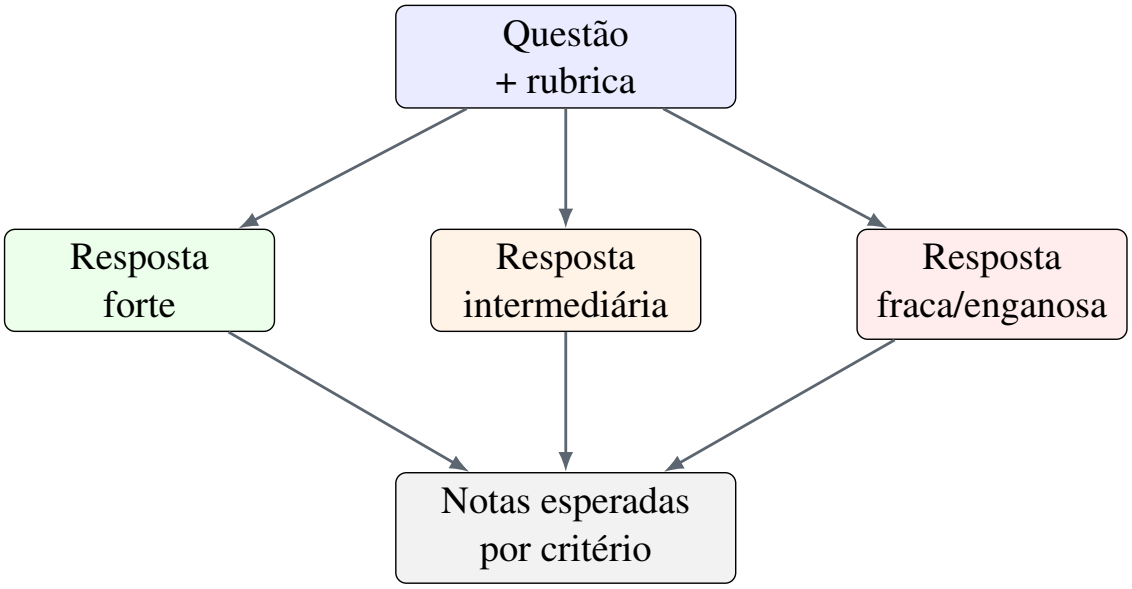

Figura 2. Estrutura conceitual das respostas de referência

Cada arquivo de referência agrupava as três respostas de uma mesma questão. A estrutura abaixo ilustra o formato, omitindo detalhes extensos de texto, código e justificativas.

Listing 1. Estrutura resumida de uma resposta de referência

```
questionSlug: questao-exemplo

answers:
  - id: questao-exemplo-strong
    profile: strong
    variant: complete_correct
    answerText: |
      ...
    code: |
      ...
    justification: |
      ...
    expectedOverall: 4.9
    expectedCriteria:
      - criterionId: criterio_principal
        expectedScore: 5.0
        rationale: |
          ...

  - id: questao-exemplo-medium
    profile: medium
    variant: partial_correct
    expectedOverall: 3.3
    expectedCriteria:
      - criterionId: criterio_principal
        expectedScore: 3.5
        rationale: |
          ...
```

```
- id: questao-exemplo-weak
  profile: weak
  variant: subtle_error
  expectedOverall: 1.4
  expectedCriteria:
    - criterionId: criterio_principal
      expectedScore: 1.5
      rationale: |
        ...
```

A nota global esperada foi derivada das notas por critério e dos pesos da rubrica. Assim, a referência humana não consistia apenas em uma nota final, mas em uma decomposição explícita da qualidade da resposta. Esse desenho foi importante porque o objetivo do experimento era medir aderência à rubrica. Uma estratégia poderia se aproximar da nota global esperada por compensação entre critérios, mas ainda assim distribuir mal os pontos internamente. Ao comparar também as notas por critério, o experimento avaliou se a estratégia respeitou a estrutura da rubrica e não apenas se chegou a uma pontuação final parecida.

7. Desenho Experimental

O experimento oficial usou 8 questões, 3 respostas de referência por questão e 3 estratégias de correção. Isso produziu 24 respostas de referência e 72 avaliações oficiais. Como as estratégias usam quantidades diferentes de chamadas ao modelo, o total operacional foi de 144 chamadas: 24 na avaliação direta, 48 na avaliação com revisão e 72 nas avaliações agregadas.

A amostra foi equilibrada por eixo: duas questões de correção de bugs, duas de qualidade de código, duas de testes e duas de arquitetura. A seleção priorizou questões com rubrica clara, solução esperada rica, potencial para respostas de diferentes níveis de qualidade e capacidade de diferenciar estratégias. Questões muito fáceis foram evitadas na amostra final porque tendem a gerar pouco espaço para distinção entre protocolos. Questões de arquitetura foram mantidas para incluir respostas textuais estruturadas e decisões de projeto, enquanto os demais eixos concentraram questões com código.

Essa amostra não pretende representar todo o universo de respostas de estudantes. Ela foi construída como benchmark interno controlado, adequado para comparar protocolos sob condições conhecidas.

8. Estratégias de Correção

Foram comparadas três estratégias oficiais de correção. No texto principal, elas são apresentadas por seus nomes conceituais em português, pois os identificadores de implementação não são relevantes para o relatório acadêmico.

8.1. Avaliação direta

A avaliação direta usa uma única chamada ao modelo. A entrada contém a questão, a rubrica, a solução esperada e a resposta avaliada. O modelo retorna uma nota global, notas por critério, evidências e feedback. Essa estratégia funciona como linha de base operacional: é a forma mais simples, barata e direta de usar o avaliador.

8.2. Avaliação com revisão

A avaliação com revisão usa duas chamadas. A primeira produz uma avaliação inicial. A segunda atua como revisora técnica, recebendo a avaliação inicial, a resposta e a rubrica. O revisor deve procurar inconsistências, omissões e desalinhamentos, preservando o que estiver correto e ajustando apenas o que a rubrica justificar.

Essa estratégia testa uma hipótese intuitiva: uma segunda leitura crítica poderia corrigir falhas da primeira. O experimento mostra, porém, que essa hipótese depende fortemente do prompt de revisão.

8.3. Avaliações agregadas

As avaliações agregadas realizam três avaliações independentes da mesma resposta, sempre com o mesmo modelo, mas com focos diferentes:

1. aderência estrita à rubrica;
2. evidências textuais presentes na resposta;
3. cobertura conceitual dos pontos esperados.

As notas finais são agregadas por critério, usando a média dos julgamentos. A nota global é então recalculada a partir dos critérios e de seus pesos. A Figura 3 compara visualmente as três estratégias.

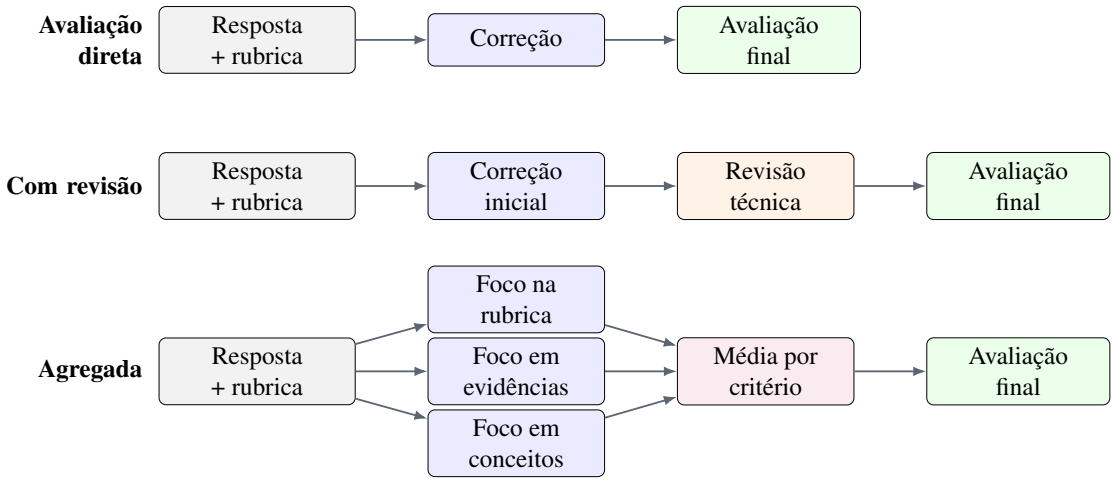

Figura 3. Representação das três estratégias de correção comparadas

9. Métricas de Avaliação

As métricas foram escolhidas para capturar tanto qualidade de correção quanto viabilidade operacional. A Tabela 9 resume cada métrica.

Tabela 9. Métricas usadas na avaliação experimental

Métrica	Interpretação
Erro absoluto médio da nota global	Mede a distância média entre a nota final prevista e a nota final esperada. Quanto menor, melhor.
Erro absoluto médio por critério	Mede a aderência fina à rubrica, comparando cada critério previsto com sua nota esperada.
Erro assinado médio	Indica tendência de generosidade ou severidade. Valor positivo indica superavaliação média; valor negativo indica subavaliação média.
Custo estimado	Mede o consumo operacional estimado pelo provedor, permitindo comparar qualidade com custo.
Número de chamadas	Quantifica quantas invocações ao modelo foram necessárias.
Latência média	Mede o tempo médio de execução por resposta avaliada.
Robustez	Registra falhas de parsing, schema, modelo ou timeout.

Formalmente, para uma resposta i, seja g_i a nota global esperada e $\hat{g}_i$ a nota global prevista. O erro absoluto global é $|g_i - \hat{g}_i|$. O erro absoluto médio global é:

$$MAE_{global} = \frac{1}{n} \sum_{i=1}^{n} |g_i - \hat{g}_i|$$

Para critérios, seja c_{ij} a nota esperada do critério j na resposta i e $\hat{c}_{ij}$ a nota prevista. O erro por critério é calculado sobre todos os critérios correspondentes:

$$MAE_{criterio} = \frac{1}{m} \sum_{i,j} |c_{ij} - \hat{c}_{ij}|$$

O erro por critério é especialmente importante porque uma nota global correta pode esconder compensações. Por exemplo, uma estratégia pode superestimar um critério e subestimar outro, chegando a uma nota final parecida com a esperada. Nesse caso, a nota global parece adequada, mas a correção não respeitou bem a rubrica.

10. Resultados Oficiais

A Tabela 10 apresenta os resultados oficiais. No conjunto de execuções analisado, não foram observadas falhas de parsing, validação de schema, chamada ao modelo ou timeout.

Tabela 10. Resultados oficiais por estratégia

Estratégia	Erro global	Erro critério	Custo	Chamadas	Latência	Erro ass.
Avaliação direta	0,2658	0,5143	0,0929	24	7,01 s	0,1400
Avaliação com revisão	0,3092	0,5190	0,1822	48	12,95 s	0,1067
Avaliações agregadas	0,2550	0,4538	0,2785	72	20,20 s	0,0983

As avaliações agregadas obtiveram o melhor desempenho oficial em aderência à rubrica. A diferença no erro global em relação à avaliação direta foi pequena, mas a diferença no erro por critério foi mais clara. Isso sugere que a agregação ajudou principalmente na distribuição da nota entre critérios, e não apenas na nota final.

A avaliação direta teve desempenho global próximo e custo muito menor. Ela usou um terço das chamadas das avaliações agregadas e apresentou latência média significativamente menor. Assim, no recorte experimental analisado, a avaliação direta foi a alternativa de melhor custo-benefício.

A avaliação com revisão não melhorou a linha de base. Ela teve erro global e erro por critério maiores que a avaliação direta, além de dobrar o número de chamadas. O resultado mostra que uma segunda chamada não garante melhoria: a revisão precisa estar bem calibrada para corrigir mais do que atrapalhar.

11. Diagnóstico dos Resultados

11.1. Avaliações agregadas

A comparação direta entre avaliação direta e avaliações agregadas foi feita em 24 pares de respostas. As avaliações agregadas venceram em 8 casos, a avaliação direta venceu em 7 e houve empate prático em 9. Portanto, a agregação melhorou o resultado agregado, mas não venceu universalmente.

Tabela 11. Comparação pareada entre avaliação direta e avaliação agregada

Resultado da comparação	Pares
Avaliação agregada mais próxima da referência	8
Avaliação direta mais próxima da referência	7
Empate prático	9
Total	24

Por perfil de resposta, o ganho da agregação foi mais claro nas respostas intermediárias. Nas respostas fortes, a avaliação direta já estava muito próxima da referência, deixando pouco espaço para melhoria. Nas respostas fracas ou enganosas, o comportamento foi mais variável, sem ganho uniforme.

Outro achado importante foi a baixa dispersão interna entre as três chamadas das avaliações agregadas. Em vários casos, as avaliações independentes concordaram entre si, mas ainda erraram em relação à referência. Isso sugere que parte do erro vinha de vieses comuns do modelo ou do prompt, e não apenas de instabilidade aleatória. Quando três avaliações erram na mesma direção, a média não consegue corrigir o problema.

11.2. Avaliação com revisão

A análise da revisão oficial comparou a avaliação inicial com a avaliação final revisada. Em 24 execuções, a revisão melhorou 4 casos, piorou 5 e manteve 15 praticamente inalterados. A Tabela 12 resume esse comportamento.

Tabela 12. Efeito da revisão oficial sobre a avaliação inicial

Efeito observado	Casos
Melhorou em relação à avaliação inicial	4
Piorou em relação à avaliação inicial	5
Manteve resultado praticamente igual	15

Esse resultado ajuda a interpretar por que a revisão oficial não compensou. A segunda chamada não falhou tecnicamente: ela produziu JSON válido e foi capaz de revisar a avaliação. O problema foi metodológico. As alterações feitas pelo revisor nem sempre aproximaram a nota da referência esperada. Em alguns casos, houve sobrecorreção: o revisor alterou uma avaliação inicial plausível e piorou a aderência à rubrica.

11.3. Versões alternativas do prompt de revisão

Após o experimento oficial, duas versões alternativas do prompt de revisão foram avaliadas como análises de sensibilidade. Essas análises não substituem o experimento oficial, pois foram feitas depois de observar o comportamento inicial. Elas servem para diagnosticar a sensibilidade da revisão ao prompt.

A versão conservadora foi instruída a preservar a nota inicial salvo em caso de erro claro. Ela teve erro global 0,2700, alterou apenas 1 das 24 notas e essa única alteração piorou o erro. Na prática, essa versão quase desligou a revisão. Ela foi melhor que a revisão oficial porque evitou intervenções ruins, mas não demonstrou ganho real sobre a avaliação direta.

A versão calibrada foi instruída a corrigir notas quando houvesse ganho claro de alinhamento com a rubrica, evitando tanto conservadorismo excessivo quanto ajustes cosméticos. Ela teve erro global 0,2146, alterou 12 das 24 notas, melhorou 8 casos, piorou 4 e manteve 12. Esse resultado foi promissor, especialmente em respostas intermediárias e fracas, mas deve ser tratado como exploratório. Como o prompt foi ajustado após diagnóstico anterior, sua validade exige teste em nova amostra.

Tabela 13. Resultados oficiais e análises exploratórias de revisão

Estratégia	Papel no estudo	Erro global	Chamadas adicionais	Custo adicional
Revisão oficial	Resultado oficial	0,3092	24	0,0893
Revisão conservadora	Sensibilidade	0,2700	24	0,0950
Revisão calibrada	Exploratória	0,2146	24	0,1005

A Tabela 13 mostra o custo adicional da etapa de revisão quando ela é aplicada sobre uma avaliação inicial já existente. Como estratégia completa, qualquer avaliação com revisão exige duas chamadas por resposta: uma para a avaliação inicial e outra para a revisão.

12. Discussão

O primeiro achado do trabalho é que avaliar respostas abertas com LLMs exige mais do que chamar um modelo. O protocolo de avaliação influencia qualidade, custo e latência. Mesmo usando o mesmo modelo e o mesmo conjunto de respostas, as estratégias produziram erros diferentes.

O segundo achado é que a avaliação direta é uma linha de base forte. Ela teve erro global próximo ao da melhor estratégia oficial e custo muito menor. Isso é importante para uso prático: em muitos cenários educacionais, uma estratégia simples, barata e suficientemente boa pode ser preferível a uma estratégia marginalmente melhor, mas três vezes mais cara.

O terceiro achado é que avaliações agregadas ajudam principalmente na aderência por critério. O ganho em erro global foi pequeno, mas o ganho no erro por critério foi mais expressivo. Isso indica que a agregação foi mais útil para distribuir melhor a pontuação entre dimensões da rubrica do que para alterar drasticamente a nota final.

O quarto achado é que revisão automática é sensível ao prompt. A revisão oficial não melhorou o resultado e teve custo maior. A versão conservadora evitou pioras apenas por quase não intervir. A versão calibrada mostrou potencial, mas precisa ser validada em nova amostra para não confundir melhoria real com ajuste pós-análise.

Por fim, o erro por critério se mostrou uma métrica mais informativa que apenas a nota global. Em correção por rubrica, o objetivo não é somente chegar a uma nota final parecida, mas justificar essa nota com base nos critérios corretos. Essa granularidade é essencial para aplicações educacionais, pois o feedback do estudante depende de quais critérios foram atendidos ou não.

13. Ameaças à Validade e Limitações

As principais limitações do estudo são:

- o conjunto de respostas de referência foi criado manualmente e não contém respostas reais de estudantes;
- a amostra experimental é pequena, com 8 questões e 24 respostas;
- as notas esperadas foram definidas por curadoria do autor, podendo refletir vieses individuais;
- apenas um modelo foi usado no experimento oficial;
- manter o modelo fixo isola o efeito do protocolo, mas limita conclusões sobre outros modelos;
- as análises de revisão conservadora e calibrada foram posteriores ao experimento oficial e precisam de validação independente;
- custo e latência dependem do provedor, do modelo, da carga do serviço e das condições de rede;
- a amostra experimental cobre apenas parte do banco final, portanto os resultados devem ser interpretados como evidência sobre o conjunto avaliado, não como exaustão de todos os cenários possíveis.

Essas limitações não invalidam o recorte experimental, mas restringem a interpretação. A conclusão adequada não é que uma estratégia é universalmente melhor. A conclusão é que, neste banco, nesta amostra e com este modelo, os protocolos apresentaram trade-offs mensuráveis.

14. Conclusão

O POC II entregou uma reorganização substancial do banco de questões, um pipeline semiautomático de criação e revisão com curadoria humana, um conjunto controlado de respostas de referência e uma comparação sistemática entre protocolos de correção automática com LLMs.

No experimento oficial, as avaliações agregadas obtiveram o menor erro por critério e o menor erro global, sendo a melhor alternativa em aderência à rubrica no recorte analisado. A avaliação direta apresentou desempenho muito próximo em nota global, com custo e latência substancialmente menores, tornando-se a alternativa de melhor custo-benefício nesse mesmo recorte. A avaliação com revisão, na versão oficial, não compensou: aumentou custo e erro em relação à linha de base.

As análises posteriores mostraram que a revisão não deve ser descartada como ideia, mas precisa de prompt bem calibrado e validação independente. A versão calibrada apresentou resultado promissor, mas por ter sido desenhada após observação dos resultados anteriores, deve ser considerada hipótese para novo experimento, não conclusão oficial.

15. Trabalhos Futuros

Como trabalhos futuros, recomenda-se:

- validar a revisão calibrada em nova amostra, sem ajuste posterior ao conjunto avaliado;
- coletar respostas reais de estudantes para medir validade externa;
- comparar diferentes modelos mantendo o mesmo protocolo de avaliação;
- estudar concordância entre LLMs e avaliadores humanos;
- testar métodos alternativos de agregação, como mediana, voto por critério ou seleção por consistência;
- combinar avaliação por LLM com testes automatizados ou análise estática em questões com código;
- investigar robustez contra respostas enganosas, respostas manipulativas e tentativas de induzir o avaliador.

A. Prompts de Avaliação e Revisão

Este apêndice registra os principais prompts usados no estudo. Para evitar reproduzir dados de uma questão específica, campos dinâmicos são apresentados como placeholders. As instruções fixas de avaliação, revisão e agregação são mantidas em forma textual.

A.1. Prompt da avaliação direta

```
[SYSTEM]
Você é um avaliador técnico especializado em Engenharia de Software.
Siga fielmente a rubrica da questão, atribuindo notas numéricas.
Responda ESTRITAMENTE com JSON válido; não escreva texto fora do JSON.
Se o aluno não respondeu, use notas mínimas e explique no feedback.

[USER]
Avalie a resposta do aluno para a questão abaixo usando a rubrica
    fornecida.
```

```
[QUESTÃO / ENUNCIADO]
{{{enunciado}}}

[RUBRICA DE AVALIAÇÃO]
{{{rubrica}}}

[PONTOS ESPERADOS NA RESPOSTA]
{{{pontos_esperados}}}

[RESPOSTA DO ALUNO]
{{{resposta}}}

[INSTRUÇÕES DE AVALIAÇÃO]
- Atribua notas de 0 a 5 por critério.
- Use evidências objetivas da resposta do aluno.
- Não premie informação que não aparece na resposta.
- Não penalize ausência de conteúdo que a rubrica não exige.
- A nota global deve ser coerente com os critérios e pesos da rubrica.

[FORMATO DE SAÍDA - JSON]
{
  "overall": 0-5,
  "criteria": [
    {
      "criterionId": "...",
      "score": 0-5,
      "weight": 0.0,
      "evidence": "...",
      "comment": "..."
    }
  ],
  "feedback": "..."
}
```

A.2. Prompt da revisão oficial

```
[SYSTEM]
Você é um revisor técnico de avaliações de Engenharia de Software.
Seu papel é revisar a avaliação inicial contra a rubrica, procurando
    inconsistências,
omissões e desalinhamentos.
Não faça uma avaliação independente do zero; preserve o que estiver
    correto e ajuste
apenas o que a rubrica justificar.
Responda estritamente com JSON válido no mesmo schema da avaliação
    normal:
overall, criteria e feedback.
Você deve comparar explicitamente: rubrica, resposta avaliada e avaliaç
    ão inicial.
Não altere notas apenas por preferência textual; qualquer ajuste deve
    ser justificado
por evidência presente ou ausente na resposta.
Se a avaliação inicial já estiver adequada, mantenha as notas e apenas
    melhore
comentários/evidências quando necessário.
```

```
Não premie informação que não aparece na resposta avaliada.
Não penalize ausência de conteúdo que a rubrica não exige.

[USER]
Tarefa: revise a avaliação inicial contra a rubrica e retorne a avaliaç
    ão final.

Entrada:
- Questão: título, eixo, dificuldade, enunciado e rubrica.
- Resposta avaliada: texto, código e/ou justificativa.
- Avaliação inicial: overall, criteria e feedback.

Requisitos:
- Revise inconsistências entre notas, evidências, comentários e rubrica
    .
- Corrija omissões relevantes se a avaliação inicial ignorou pontos da
    resposta.
- Mantenha a escala 0-5.
- Retorne somente JSON válido no schema exigido.
- Para cada critério, verifique se a nota atribuída é compatível com a
    evidência.
- Preserve notas corretas da avaliação inicial.
- Altere apenas quando houver inconsistência clara com a rubrica.
- Não introduza novos critérios e não remova critérios da rubrica.
- Garanta que todos os critérios da rubrica apareçam em criteria.
```

A.3. Prompt da revisão conservadora

```
[SYSTEM]
Você é um revisor técnico CONSERVADOR de avaliações de Engenharia de
    Software.
Seu papel não é reavaliar a resposta do zero, mas auditar se a avaliaçã
    o inicial
contém erro claro em relação à rubrica.
Seu padrão deve ser PRESERVAR as notas da avaliação inicial.
Altere uma nota apenas quando houver incompatibilidade clara e
    relevante entre
a rubrica, a resposta avaliada e a nota inicial.
Em casos ambíguos, limítrofes ou discutíveis, mantenha a nota inicial.
Não altere notas por preferência textual, estilo de explicação, formula
    ção alternativa
ou detalhe marginal.
Não premie informação que não aparece explicitamente na resposta
    avaliada.
Não penalize ausência de conteúdo que a rubrica não exige
    explicitamente.
Não transforme a revisão em uma avaliação independente.
Se a avaliação inicial estiver plausível dentro da rubrica, mantenha a
    nota mesmo que
você escreveria um comentário diferente.
Evite ajustes maiores que 0.5 ponto por critério, salvo quando a avalia
    ção inicial
contrariar diretamente a rubrica ou ignorar uma evidência central.
Se alterar uma nota, explique no comentário do critério qual erro claro
     justificou
a alteração.
```

```
O campo overall deve ser coerente com a média ponderada dos critérios
    revisados.
Responda estritamente com JSON válido no mesmo schema da avaliação
    normal.

[POLÍTICA DE DECISÃO]
- Primeiro, verifique se cada nota inicial é plausível segundo a
    rubrica.
- Se a nota inicial for plausível, mantenha o score.
- Só altere o score se houver erro claro.
- Não ajuste score apenas para refinar calibragem fina.
- Não ajuste score apenas porque a resposta poderia ser mais completa,
    se a nota
  inicial já refletia essa limitação.
- Em dúvida, preserve a nota inicial.
```

A.4. Prompt da revisão calibrada

```
[SYSTEM]
Você é um revisor técnico CALIBRADO de avaliações de Engenharia de
    Software.
Seu objetivo é melhorar a acurácia da avaliação inicial contra a
    rubrica, sem
transformar a revisão em uma avaliação independente do zero.
Use a avaliação inicial como ponto de partida e preserve notas que
    estejam bem
sustentadas.
Mas corrija notas quando a resposta avaliada e a rubrica indicarem uma
    nota
materialmente melhor, mesmo que a nota inicial seja apenas plausível.
Procure tanto superestimações quanto subestimações.
Não seja conservador a ponto de manter erro corrigível; também não faça
     ajustes
cosméticos.
Ajustes pequenos de calibragem fina devem ser evitados; prefira alterar
     quando a
diferença esperada for de pelo menos 0.25 ponto no critério.
Evite ajustes maiores que 1.0 ponto por critério salvo quando houver
    evidência
muito clara.
Não premie informação que não aparece explicitamente na resposta
    avaliada.
Não penalize ausência de conteúdo que a rubrica não exige
    explicitamente.
Se alterar uma nota, explique no comentário do critério qual evidência
    da resposta
e qual ponto da rubrica justificam a mudança.
O campo overall deve ser coerente com a média ponderada dos critérios
    revisados.
Responda estritamente com JSON válido no mesmo schema da avaliação
    normal.

[POLÍTICA DE DECISÃO]
- Para cada critério, identifique primeiro a evidência explícita
    presente na resposta.
- Compare essa evidência com os descritores e pesos da rubrica.
```

```
- Compare então a nota inicial com a nota mais bem sustentada pela
    rubrica.
- Mantenha a nota inicial se ela estiver próxima da nota melhor
    sustentada.
- Altere a nota se a resposta e a rubrica sustentarem uma correção
    material.
- Não altere nota apenas por estilo, preferência textual, comentário
    incompleto
  ou diferença marginal.
- Não use expectativa externa; decida apenas por resposta, rubrica e
    avaliação inicial.
```

A.5. Prompts das avaliações agregadas

```
[VARIAÇÃO 1 - ADERÊNCIA À RUBRICA]
Nesta avaliação, priorize a aderência estrita à rubrica. Atribua os
    scores
considerando cuidadosamente os critérios, descritores e pesos definidos
    .
Penalize respostas que pareçam boas de forma geral, mas que não atendam
explicitamente ao que o critério avalia. Não use critérios externos à
    rubrica.

[VARIAÇÃO 2 - EVIDÊNCIAS TEXTUAIS]
Nesta avaliação, priorize evidências textuais presentes na resposta
    avaliada.
Ao atribuir os scores, considere principalmente o que está
    explicitamente
demonstrado no texto. Evite inferir conhecimento implícito quando ele n
    ão estiver
sustentado por trechos, argumentos, exemplos ou explicações presentes
    na resposta.

[VARIAÇÃO 3 - COBERTURA CONCEITUAL]
Nesta avaliação, priorize a cobertura conceitual da resposta. Verifique
     se os
conceitos centrais esperados foram abordados de maneira correta e
    suficiente,
mesmo que a organização, redação ou terminologia não sejam ideais.
    Penalize
omissões conceituais relevantes e confusões entre conceitos importantes
    .
```

Referências

[1] Burrows, S., Gurevych, I., Stein, B. (2015). *The Eras and Trends of Automatic Short Answer Grading*. International Journal of Artificial Intelligence in Education, 25(1), 60–117.

[2] Chiang, C.-H., Lee, H.-Y. (2023). *Can Large Language Models Be an Alternative to Human Evaluations?* Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics, 15607–15631. Disponível em: https://aclanthology.org/2023.acl-long.870/.

[3] Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., Stoica, I. (2023). *Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena*. arXiv:2306.05685. Disponível em: https://arxiv.org/abs/2306.05685.

[4] Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., Zhu, C. (2023). *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment*. Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2511–2522. Disponível em: https://aclanthology.org/2023.emnlp-main.153/.

[5] Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E., Narang, S., Chowdhery, A., Zhou, D. (2023). *Self-Consistency Improves Chain of Thought Reasoning in Language Models*. The Eleventh International Conference on Learning Representations. Disponível em: https://arxiv.org/abs/2203.11171.

[6] Wang, P., Li, L., Chen, L., Cai, Z., Zhu, D., Lin, B., Cao, Y., Liu, Q., Liu, T., Sui, Z. (2024). *Large Language Models are not Fair Evaluators*. Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics, 9440–9450. Disponível em: https://arxiv.org/abs/2305.17926.

[7] Chamieh, I., Zesch, T., Giebermann, K. (2024). *LLMs in Short Answer Scoring: Limitations and Promise of Zero-Shot and Few-Shot Approaches*. Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications, 309–315. Disponível em: https://aclanthology.org/2024.bea-1.25/.

[8] Xie, W., Niu, J., Xue, C. J., Guan, N. (2024). *Grade Like a Human: Rethinking Automated Assessment with Large Language Models*. arXiv:2405.19694. Disponível em: https://arxiv.org/abs/2405.19694.

[9] Chiang, C.-H., Chen, W.-C., Kuan, C.-Y., Yang, C., Lee, H.-Y. (2024). *Large Language Model as an Assignment Evaluator: Insights, Feedback, and Challenges in a 1000+ Student Course*. Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, 2556–2571. Disponível em: https://aclanthology.org/2024.emnlp-main.146/.