

Mitigando Alucinações no Ensino de Lógica de Programação: Uma Avaliação de Agentes Conversacionais Baseados em RAG

Igor Santos,¹ Martiniano Gomes,¹ Vitória Leda,¹ Marcus Coelho,¹
Ramon Barbosa,¹ Aline Valente,² Warley Junior,¹ Elton Alves¹

¹Instituto de Geociências e Engenharias - Unifesspa - Marabá, PA - Brasil

²Programa de Pós-Graduação em Computação - UFPA - Belém, PA - Brasil

{igor.correa, martinianogomes, wmvj}@unifesspa.edu.br

Abstract. *Conversational agents in programming education are often hindered by language model hallucinations. Although Retrieval-Augmented Generation (RAG) resolves this by anchoring responses in instructional materials, determining which architecture best balances cost and accuracy remains an open challenge. To address this gap, this study conducted a benchmarking of six RAG approaches in programming logic instruction. The results show that single-step retrieval models (Context and Self RAG) achieved the highest faithfulness and semantic relevance, consuming only 1,200 tokens in 4 seconds. We conclude that simpler RAG architectures outperform graph-based approaches, successfully uniting pedagogical precision with operational efficiency.*

Resumo. *Agentes conversacionais no ensino de programação sofrem com alucinações dos modelos de linguagem. Embora o RAG resolva isso ancorando respostas em materiais didáticos, resta definir qual arquitetura equilibra melhor custo e precisão. Para preencher essa lacuna, este estudo realizou um benchmarking de seis abordagens RAG no ensino de lógica. Os resultados mostram que modelos de passo único (Context e Self RAG) atingiram maior fidelidade e relevância semântica, consumindo apenas 1200 tokens em 4 segundos. Conclui-se que arquiteturas RAG simples superam abordagens em grafos, unindo precisão pedagógica e eficiência operacional.*

1. Introdução

Programadores iniciantes frequentemente enfrentam dificuldades na compreensão da lógica de programação, o que pode comprometer a motivação e resultar em frustração ou até evasão. Nesse cenário, agentes conversacionais têm se destacado como uma alternativa promissora, pois possibilitam oferecer suporte escalável e personalizado, adaptando-se às necessidades individuais dos estudantes, fornecendo *feedback* em tempo real e estimulando uma aprendizagem mais ativa [Barzanji and Loitsch 2025].

Entretanto, a utilização desses modelos levanta desafios significativos relacionados à confiabilidade das respostas geradas. Grandes Modelos de Linguagem ou *Large Language Model* (LLM) podem produzir respostas com aparência coerente, mas sem fundamentação em fontes verificáveis, caracterizando a alucinação. Esse problema torna-se particularmente crítico no domínio educacional, no qual a propagação de informações imprecisas pode impactar diretamente o processo de aprendizagem [Petrin and Néto 2026].

Uma abordagem promissora para mitigar esses desafios é o uso de técnicas de *Retrieval-Augmented Generation* (RAG), que permitem fundamentar as respostas dos modelos em fontes externas. Essa abordagem garante maior fidelidade ao conteúdo do curso ao basear as respostas em materiais didáticos específicos, além de reduzir significativamente as alucinações típicas de modelos de linguagem. Outro ponto relevante é a possibilidade de atualização contínua do conhecimento, permitindo a incorporação de novos conteúdos sem a necessidade de treinar o modelo novamente. Adicionalmente, o RAG possibilita maior transparência e rastreabilidade das respostas, ao indicar as fontes utilizadas, o que favorece a verificação e o aprofundamento do aprendizado [Borges 2025].

O restante deste artigo está organizado da seguinte forma: A fundamentação teórica é descrita na seção 2. Os trabalhos relacionados são discutidos na seção 3 e a seção 4 apresenta a metodologia da proposta. Na seção 5 é dissertado os parâmetros de configuração juntamente com os resultados adquiridos. Finalmente, a seção 6 conclui o artigo e aponta direções futuras que podem ser consideradas.

2. Fundamentação Teórica

Para melhor compreensão do comportamento de cada arquitetura de RAG na avaliação deste artigo, os parágrafos seguintes definem e correlacionam cada tecnologia, de modo a destacar a evolução do desenvolvimento destas soluções.

O Hybrid RAG (HR) [Sarmah et al. 2024] atua como a fundação estrutural ao combinar buscas lexicais (palavras-chave) e semânticas (vetores), garantindo uma recuperação inicial muito mais robusta. A partir dessa base, o Self-RAG (SR) [Asai et al. 2024] e o Memory-Augmented RAG (MAR) [Yuan et al. 2025] representam os avanços iniciais focados na autonomia do agente e na consciência do sistema. O SR inova ao treinar o modelo para buscar dados sob demanda e criticar suas próprias respostas através de tokens de autorreflexão. De maneira muito semelhante, o MAR atua focado na consciência do sistema, mas voltado para a retenção de memória global de longo prazo. Ambas as abordagens se correlacionam profundamente por transferirem o controle para a própria IA: enquanto o SR avalia criticamente o contexto da geração imediata, o MAR garante que o histórico de operações não seja perdido em tarefas prolongadas.

Evoluindo para a estruturação complexa de dados, o Graph RAG (GR) [Das and Cheng 2025] utiliza LLMs para extrair e construir grafos a partir de textos não estruturados, conectando conceitos difusos. Como uma progressão direta para cobrir perguntas ainda mais complexas, o Knowledge-Enhanced RAG (KER) [Sun et al. 2025] integra subgrafos inteiros ao pipeline de recuperação. O KER e o GR compartilham a semelhança fundamental de utilizar estruturas em rede em vez de depender puramente de vetores tradicionais. Ambos atuam para superar limitações esquemáticas, permitindo respostas mais precisas sobre bases de dados narrativas por meio de raciocínio lógico focado.

Representando o estado da arte mais moderno, o Context RAG (CR) [Prosvirnin et al. 2026] surge como uma evolução topológica desenhada para otimizar os altos custos computacionais das abordagens anteriores. Ele constrói um índice de recuperação em grafo hierárquico sem depender de LLMs para a extração exaustiva de entidades e relações. Estruturalmente, o CR compartilha o DNA baseado em redes do GR e do KER, focando em conexões complexas. No entanto, ele se destaca e se moderniza

por ser uma solução livre de extração ("extraction-free"), tornando o uso de grafos para geração aumentada muito mais eficiente e escalável.

3. Trabalhos Relacionados

A integração da arquitetura de Geração Aumentada por Recuperação (RAG) com Grandes Modelos de Linguagem (LLMs) tem se consolidado como uma estratégia fundamental para mitigar alucinações e ancorar o conhecimento em domínios educacionais específicos. O sistema CodeQA [Ahmed et al. 2024], por exemplo, demonstra alta eficácia ao processar consultas complexas no ensino de desenvolvimento de código por meio da recuperação dinâmica de documentos, embora sua dependência das nuances deste domínio exija adaptações para outras áreas. De maneira análoga, abordagens focadas em materiais acadêmicos, como as propostas por [Wu and Tseng 2025] para bases de dados universitárias e por [Alario-Hoyos et al. 2024] para cursos MOOC, garantem a aderência das respostas ao contexto da disciplina. Contudo, essas soluções enfrentam desafios operacionais significativos; enquanto [Wu and Tseng 2025] aponta para o alto custo de infraestrutura local (uso intensivo de GPUs) e limitações na janela de contexto, exigindo extração precisa via OCR, o sistema de [Alario-Hoyos et al. 2024] utiliza poda dinâmica de tokens, mas compartilha a vulnerabilidade de depender de bases de dados estáticas, demandando um esforço administrativo e manual contínuo para evitar a obsolescência do conteúdo.

No que tange à aplicação dessas tecnologias no apoio direto ao aluno, destaca-se a eficácia dos tutores virtuais focados no método socrático, que auxiliam na resolução de problemas sem fornecer respostas diretas. O RAGMan [Ma et al. 2024] exemplifica essa abordagem ao utilizar instruções rígidas para guiar estudantes em tarefas de programação, resultando em uma precisão de 98% dentro do escopo esperado e em uma redução estatisticamente significativa na taxa de reprovação, além de criar um ambiente livre de julgamentos. Compartilhando dessa mesma filosofia, o tutor de código aberto descrito em [Lin et al. 2025] foca no estímulo ao pensamento crítico com vantagens em escalabilidade e baixo custo. Apesar do impacto pedagógico positivo, ambos os sistemas apresentam barreiras de usabilidade: a adoção orgânica do RAGMan foi limitada a cerca da metade dos estudantes matriculados e sua utilidade decaiu fora do escopo designado [Ma et al. 2024], ao passo que a ferramenta de [Lin et al. 2025] enfrenta fragmentação de contexto ao processar instruções visuais simultâneas e sofre com a ausência de suporte multilíngue nativo, o que pode comprometer a fluência das interações.

Além do suporte reativo, pesquisas recentes têm avançado para a modelagem preditiva e a avaliação automatizada do desempenho discente, buscando um alinhamento mais profundo com metodologias ativas e o desenvolvimento do raciocínio algorítmico. O framework proposto por [Das and Cheng 2025] emprega grafos e agrupamento lógico fuzzy para prever trajetórias de aprendizagem, oferecendo alta interpretabilidade ao alinhar a avaliação dinâmica da proficiência a teorias cognitivas. De forma complementar, o sistema SageJavon [Zhao et al. 2026] adota um ciclo proativo ("Aprender-Praticar-Avaliar-Apoiar"), utilizando LLMs para a avaliação automatizada de códigos via métrica FUPS-Score e recomendação personalizada de exercícios. No entanto, a sofisticação dessas arquiteturas traz contrapartidas técnicas: a abordagem baseada em grafos [Das and Cheng 2025] possui escalabilidade limitada frente à inserção de novas variáveis e depende de mapeamentos pré-definidos que restringem sua rápida generalização, enquanto o módulo heurístico do SageJavon [Zhao et al. 2026] apresenta instabilidade e

baixa precisão na ausência da recuperação do RAG, falhando ao guiar alunos em tarefas mais complexas sem esse referencial de contexto.

Diante dos desafios de custo computacional, escalabilidade e generalização evidenciados pelas diversas arquiteturas, a otimização empírica dos modelos torna-se uma etapa crucial para viabilizar a implantação de ferramentas de IA na educação. O estudo de [Yang et al. 2025] aborda diretamente esse gargalo ao avaliar métodos de busca através do framework RAGAS, comprovando que o uso de prompts estruturados e técnicas de re-ranqueamento permite que modelos leves e de código aberto alcancem um desempenho equivalente ao de modelos comerciais de alto custo. Contudo, a validação dessa otimização restringe-se a um corpus de apenas 280 amostras em uma única disciplina, o que reduz sua robustez estatística e a imediata transferibilidade para outras áreas, ressaltando a necessidade convergente em todos os estudos avaliados de ampliar as pesquisas e as bases de teste para consolidar essas inovações pedagógicas.

Diferentemente dos trabalhos anteriores que propõem arquiteturas singulares com limitações de escalabilidade (como os grafos rígidos em [Das and Cheng 2025]) ou que demandam alto custo de infraestrutura ([Wu and Tseng 2025]), este trabalho contribui ao preencher a lacuna metodológica da otimização de sistemas tutores por meio de um benchmarking empírico e comparativo de seis abordagens distintas de RAG (Context, Graph, Hybrid, KE, Memory-Augmented e Self-RAG) aplicadas especificamente ao ensino de lógica de programação. Enquanto o estudo de [Yang et al. 2025] limitou-se a uma validação em um conjunto reduzido de aprendizado de máquina, esta pesquisa avança ao utilizar os frameworks RAGAS e LangSmith para avaliar não apenas a fidelidade e precisão do contexto, mas também o custo operacional prático em termos de latência e consumo de tokens. A principal contribuição do trabalho reside na comprovação empírica de que arquiteturas de menor complexidade estrutural e com recuperação em passo único, superam as abordagens mais complexas baseadas em grafos no domínio do ensino de programação.

4. Proposta Metodológica

A Figura 1 apresenta a proposta metodológica utilizada para a avaliação de desempenho das arquiteturas RAG aplicadas ao contexto do ensino de lógica de programação.

Inicialmente, foram implementadas seis abordagens distintas de RAG com o objetivo de realizar o *benchmarking* experimental. Em seguida, realizou-se a coleta de dez materiais didáticos cujo conteúdo cobre assuntos sobre lógica booleana tradicional e lógica de programação, como livros e apostilas digitais provenientes de Instituições de Ensino Superior (IES) e Institutos Federais de Educação, Ciência e Tecnologia (IFES), de diferentes regiões do País (sul, sudeste, nordeste, norte e etc). Após a etapa de coleta, os conteúdos passaram por um processo de limpeza e filtragem, removendo capas, contracapas, páginas de agradecimentos, referências e informações impróprias ou irrelevantes para garantir maior qualidade da base documental utilizada e redução do consumo de tokens com conteúdos que não agregam na formulação de uma resposta feita pela LLM. Posteriormente, LLMs foram configurados com prompts específicos para gerar automaticamente dez pares de perguntas e respostas (veja a tabela 2). O objetivo desta etapa foi garantir que o material cobrisse diversos tipos de questões possíveis (nível de dificuldade, fácil, média e difícil), simulando de forma fiel os cenários reais de aprendizagem e os testes práticos enfrentados pelos alunos.

Próxima etapa e a mais crítica por garantir a confiabilidade técnica foi a curadoria humana. A curadoria humana foi conduzida por um professor com experiência no ensino de lógica de programação, responsável pela criação e validação do conjunto de perguntas e respostas com base no banco de materiais didáticos coletado. Essa etapa seguiu quatro critérios principais: (1) fidelidade conceitual ao conteúdo das obras de referência; (2) adequação ao nível de dificuldade (fácil, média ou difícil), definida a partir da própria vivência do professor em sala de aula e de sua análise das dificuldades recorrentes observadas em seus alunos ao longo do estudo de lógica de programação; (3) clareza e objetividade na redação de perguntas e respostas; e (4) representatividade equilibrada dos principais tópicos do domínio. Esse processo assegurou que o gabarito de referência utilizado nas métricas do RAGAS apresentasse padrão de correção técnica e pedagógica confiável, refletindo não apenas o conteúdo curricular das instituições de ensino que compuseram a base documental, mas também as dificuldades práticas enfrentadas por estudantes reais no aprendizado da disciplina.

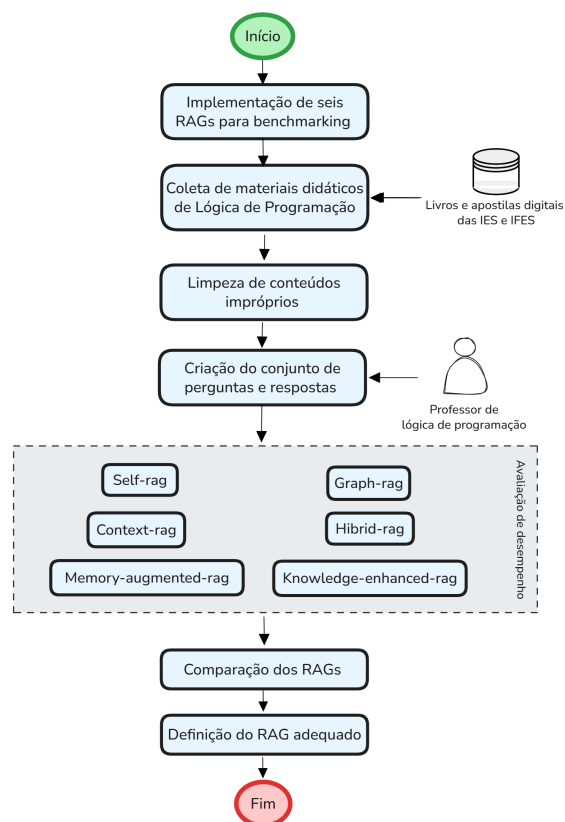


Figura 1. Metodologia de avaliação de desempenho dos RAGs com materiais de lógica de programação.

Por fim, os materiais e o conjunto de perguntas e respostas foram submetidos às arquiteturas *Self RAG*, *Context RAG*, *Memory Augmented RAG*, *Graph RAG*, *Hybrid RAG* e *Knowledge Enhanced RAG*, permitindo a avaliação comparativa do desempenho de cada modelo, considerando quatro métricas relacionadas a qualidade de resposta e contexto dos RAGs e duas métricas relacionadas aos custo operacional dos RAGs. Os resultados obtidos foram analisados para realizar a comparação entre os RAGs e definir a arquitetura

de RAG mais adequada para aplicação em ferramentas de apoio ao ensino de lógica de programação.

4.1. Métricas de desempenho dos RAGs

Nesta pesquisa, escolhemos quatro indicadores específicos derivados do framework RAGAS [Es et al. 2024] para avaliar a qualidade das informações recuperadas e a qualidade das respostas sob várias perspectivas, que incluem cobertura de informações, consistência de entidades, relevância e correção de fatos, e dois indicadores derivados da plataforma LangSmith [LangChain, Inc. 2024] para avaliar a quantidade total de tokens consumidos e o tempo de resposta para cada modelo de RAG. As definições e as etapas processuais envolvidas em cada indicador são fornecidas abaixo:

1. A **Fidelidade (*Faithfulness*)** avalia a consistência factual de uma resposta gerada em relação ao contexto recuperado. É definida como:

$$F = \frac{SC_R}{TC_R} \quad (1)$$

Aqui, SC_R é o número de afirmações na resposta que são suportadas pelo contexto, e TC_R representa o número total de afirmações na resposta. Para calcular essa métrica, a resposta gerada é primeiro dividida em afirmações factuais individuais usando um modelo de linguagem. Em seguida, cada afirmação é avaliada em relação ao contexto recuperado para determinar se pode ser inferida ou fundamentada. Uma afirmação é classificada como suportada se o contexto recuperado contiver informações que sejam semanticamente consistentes ou logicamente implicadas. Esta etapa garante que a resposta evite incluir informações alucinadas ou infundadas. Pontuações de fidelidade mais elevadas indicam um maior alinhamento factual com o contexto de suporte.

2. A **Relevância da Resposta (*Response Relevance*)** mede a consistência semântica entre a resposta gerada e a pergunta original. É calculada como:

$$RR = \frac{1}{N} \sum_{i=1}^N \cos(E_{gi}, E_o) \quad (2)$$

onde E_{gi} é o *embedding* da i -ésima pergunta gerada reversamente e E_o é o *embedding* da entrada original do usuário. Para calcular essa métrica, o modelo inicialmente cria um conjunto de perguntas geradas com base na resposta. As perguntas visam encapsular as informações mais importantes da resposta. *Embeddings* de frases são então gerados para todas as perguntas geradas e para a consulta do usuário. A similaridade de cosseno de cada par é calculada e depois feita a média para derivar a pontuação final. Valores altos demonstram uma maior relação semântica e maior relevância da resposta para a consulta do usuário.

3. A **Precisão de Contexto (*Context Precision*)** no rank K é uma medida que calcula a relevância dos Top-K contextos recuperados em relação a uma resposta de referência e à entrada do usuário. Para descobrir a relevância de cada bloco (chunk) recuperado individualmente, a resposta de referência é comparada diretamente usando um modelo de linguagem. Sua medição é expressa abaixo:

$$CP = \frac{\sum_{i=1}^K (P@k \times v_k)}{Relevant@K} \quad (3)$$

$$P@k = \frac{TP_k}{TP_k + FP_k} \quad (4)$$

Aqui, $P@k$ é a precisão no rank k , $v_k \in \{0,1\}$ indica a relevância, e $Relevant@K$ é o número de contextos relevantes dentro dos K principais resultados. Para pontuar a relevância, cada bloco recuperado é avaliado em relação à resposta de referência por um modelo de linguagem. Um bloco é considerado relevante se contiver informações semanticamente próximas ou que suportem a resposta de referência. A avaliação é feita com base em *prompts* estruturados para manter a consistência entre as amostras. Altas pontuações de CP são um sinal de um ranqueamento superior de conteúdo relevante nas primeiras posições de recuperação.

4. A **Revocação de Contexto** (*Context Recall*) calcula o número de afirmações relevantes de uma resposta de referência que são de fato cobertas pelo contexto recuperado, enfatizando a completude da recuperação. Este indicador garante a preservação das informações-chave da fonte de referência. É dado por:

$$CR = \frac{\text{Afirmações Fundamentadas}}{\text{Total de afirmações}} \quad (5)$$

A resposta de referência original é decomposta em afirmações factuais discretas por um modelo de linguagem. Cada uma das afirmações é então verificada dentro do contexto recuperado para determinar seu nível de suporte. Uma afirmação é considerada fundamentada se o contexto contiver informações semanticamente equivalentes ou parafraseadas que reflitam corretamente a afirmação. Dessa forma, todas as informações necessárias são verificadas. Um valor alto de CR significa uma alta cobertura de recuperação.

5. O **total de tokens consumidos** registra os valores fornecidos pelo provedor de modelo (como OpenAI ou Anthropic) através do campo "usage_metadata" do LangSmith. É calculada como:

$$T_{\text{total}} = T_{\text{entrada}} + T_{\text{saída}} \quad (6)$$

onde T_{total} é o número total de tokens consumidos na operação, T_{entrada} corresponde a tudo o que é enviado ao modelo. Isso inclui a sua pergunta, as instruções de sistema (*system prompt*), o histórico da conversa e os textos de documentos anexados, e $T_{\text{saída}}$ corresponde exclusivamente ao texto gerado pelo RAG como resposta.

6. O **tempo total de resposta (Latência)** é calculado diretamente pelo LangSmith com base em *timestamps* internos. É calculada como:

$$\text{Latência End-to-End} = t_{\text{rede}} + \text{TTFT} + [\text{TPOT} \times (N - 1)] \quad (7)$$

onde t_{rede} é a latência de rede (ida e volta do pacote HTTP/WebSocket), que inclui tempo de DNS, estabelecimento de conexão TLS e roteamento, TTFT (*Time To First Token*) é o tempo que o modelo leva para processar todo o texto de entrada e gerar o primeiríssimo token da resposta. Este valor depende fortemente

do tamanho do *prompt* enviado, TPOT (*Time Per Output Token*) é o tempo médio necessário para o modelo gerar cada token subsequente (fase conhecida como *decoding*), e N é o número total de tokens gerados na resposta. A fórmula utiliza $(N - 1)$ porque o tempo gasto para gerar o primeiro token já está integralmente contabilizado na métrica TTFT.

5. Avaliação e discussão dos Resultados

Esta seção detalha os parâmetros de configuração adotados nas seis arquiteturas de RAG implementadas neste trabalho, bem como os resultados de desempenho dos RAGs baseado nas métricas apresentadas na seção anterior.

A Tabela 1 estabelece as diretrizes globais para o Grande Modelo de Linguagem (LLM), adotando uniformemente o modelo gpt-5.5 da OpenAI. A manutenção da temperatura no valor padrão do modelo (0.7) representa uma decisão técnica importante, visto que esse hiperparâmetro controla o grau de aleatoriedade na geração de texto. Ao evitar temperaturas elevadas, o sistema prioriza respostas mais determinísticas, conservadoras e factualmente ancoradas nos materiais didáticos, o que é fundamental para mitigar alucinações no ensino de programação.

Tabela 1. Parâmetros de Configuração do LLM

Parâmetro	Valor / Descrição
Modelo de Linguagem	ChatOpenAI
Nome do Modelo	gpt-5.5
Temperatura	0.7 (padrão do modelo)

O link disponível em <https://github.com/RSPLE>, explicita todos os detalhes técnicos da avaliação e do repositório com todos os materiais didáticos utilizados. Informações sobre prompts, chunking, tamanho dos chunks, overlap, embeddings, banco vetorial, configuração dos grafos, parâmetros de recuperação, versões das bibliotecas e pipeline completo de cada arquitetura de RAG.

A Tabela 2 apresenta o conjunto de perguntas e respostas conforme a metodologia apresentada na seção 4. Para cada par de pergunta/resposta, cada modelo de RAG foi avaliado trinta vezes repetidamente para que se tenha um intervalo de confiança nos ciclos de execuções de cada RAG.

A Figura 2 apresenta os resultados da distribuição de tokens e tempo de raciocínio por modelo RAG. Observa-se que modelos com abordagem baseada em grafos consomem mais tokens (em média 4900 tokens) e exigem um maior tempo de resposta (em média 12 segundos), como é o caso do *Graph Rag* e *Knowledge Enhanced Rag*. Agora, modelos como o *Context Rag*, *Hybrid Rag* e *Self Rag* obtiveram bons resultados, com baixo custo de tokens (em média 1200 tokens) e baixo tempo de resposta (em média 4 segundos), e um dos motivos é a recuperação em passo único buscando contexto uma única vez antes de gerar a resposta, diferente do *Graph Rag* e *Knowledge Enhanced Rag* que fazem buscas em múltiplas fontes, como: vetor, grafo e ferramentas.

Tabela 2. Lista de perguntas e respostas em diferentes níveis de dificuldade.

Nível	Id	Pergunta	Resposta ideal
Fácil	Q1	O que significa 'lógica de programação' em palavras simples?	Lógica de programação é o uso correto das leis do pensamento, da 'ordem da razão' e de processos formais de raciocínio e simbolização na programação de computadores, com o objetivo de produzir soluções logicamente válidas e coerentes para resolver problemas.
	Q2	De um jeito bem direto: o que é um algoritmo?	Um algoritmo é uma sequência de passos bem definidos que têm por objetivo solucionar um determinado problema.
	Q3	Qual é a diferença entre constante e variável?	Um dado é constante quando não sofre variação durante a execução do algoritmo: seu valor permanece constante do início ao fim (e também em execuções diferentes ao longo do tempo). Já um dado é variável quando pode ser alterado em algum instante durante a execução do algoritmo, ou quando seu valor depende da execução em um certo momento ou circunstância.
	Q4	Pra que serve o comando 'leia' em um algoritmo?	O comando de entrada de dados 'leia' é usado para que o algoritmo receba os dados de que precisa: ele tem a finalidade de atribuir o dado fornecido à variável identificada, seguindo a sintaxe <code>leia(identificador)</code> (por exemplo, <code>leia(X)</code> ou <code>leia(A, XPTO, NOTA)</code>).
Média	Q5	O que é um comando de atribuição e por que o tipo do dado precisa ser compatível com o tipo da variável?	Um comando de atribuição permite fornecer um valor a uma variável. O tipo do dado atribuído deve ser compatível com o tipo da variável: por exemplo, só se pode atribuir um valor lógico a uma variável declarada como do tipo lógico.
	Q6	O que são operadores aritméticos (como +, -, * e /) e pra que eles servem?	Operadores aritméticos são o conjunto de símbolos que representam as operações básicas da matemática (por exemplo: + para adição, - para subtração, * para multiplicação e / para divisão). Para potenciação e radiciação, o livro indica o uso das palavras-chave <code>pot</code> e <code>rad</code> .
	Q7	Pra que servem os operadores relacionais numa expressão?	Operadores relacionais são usados para realizar comparações entre dois valores de mesmo tipo primitivo. Esses valores podem ser constantes, variáveis ou expressões aritméticas, e esses operadores são comuns na construção de equações.
Difícil	Q8	O que é uma 'expressão lógica'?	Uma expressão lógica é aquela cujos operadores são lógicos ou relacionais e cujos operandos são relações, variáveis ou constantes do tipo lógico.
	Q9	Em uma repetição, o que é um contador e como ele é incrementado?	Um contador é um modo de contagem feito com a ajuda de uma variável com um valor inicial, que é incrementada a cada repetição. Incrementar significa somar um valor constante (normalmente 1) a cada repetição.
	Q10	Como funciona a repetição 'repita ... até' e o que ela garante sobre a execução do bloco?	A estrutura de repetição 'repita ... até' permite que um bloco (ou ação primitiva) seja repetido até que uma determinada condição seja verdadeira. Pela sintaxe da estrutura, o bloco é executado pelo menos uma vez, independentemente da validade inicial da condição.

A figura 3a aborda a distribuição de *Faithfulness* por RAG. De acordo com os gráficos é possível notar que os modelos *Context Rag*, *Self Rag* e *Hybrid Rag* tiveram a maior fidelidade nas respostas (em média 1.0), enquanto os modelos *Graph Rag* e *Knowledge Enhanced Rag* obtiveram resultados medianos (em média 0.6). Apesar dos modelos com melhores resultados citados anteriormente utilizarem-se de métodos de recuperação em um único passo, a fidelidade das respostas foram excelentes em comparação aos modelos com múltiplas etapas de recuperação. Dessa forma, pode-se concluir que, no caso experimental utilizado no projeto, arquiteturas que utilizam-se de múltiplos métodos de recuperação são mais robustos que o necessário para o banco de materiais separados para

os testes.

Na Figura 3b mostra-se a distribuição de relevância da resposta por RAG, nesse comparativo os modelos ficaram equiparados e com ótimos resultados (em média 0.82), destacando-se também que não houveram modelos com resultados relativamente melhores que os outros, como ocorreu nos resultados sobre a métrica da fidelidade. Isso significa que, apesar do desempenho e custo de cada modelo RAG diversificarem, essa análise mostra que a relevância da resposta não é afetada pelo método de recuperação da arquitetura experimental.

Na Figura 3c observa-se a distribuição da métrica de precisão do contexto por RAG. Diferentemente da métrica de fidelidade, os modelos apresentaram resultados bastante próximos e elevados, indicando que, de forma geral, todos foram capazes de recuperar contextos relevantes para as respostas geradas. Destacam-se os modelos *Graph RAG* e *Hybrid RAG*, nos quais a maior parte das execuções atingiu precisão igual a 1.0, enquanto os demais modelos permaneceram concentrados entre 0.88 e 1.0. Esse comportamento pode ser explicado pelo fato de que arquiteturas mais complexas, como *Graph RAG* e *Hybrid RAG*, utilizam múltiplas estratégias de recuperação e expansão semântica, aumentando a probabilidade de recuperar trechos altamente relacionados à consulta. Assim, mesmo quando ocorre instabilidade em métricas como *Faithfulness* ou *Recall*, os contextos efetivamente recuperados tendem a ser semanticamente precisos.

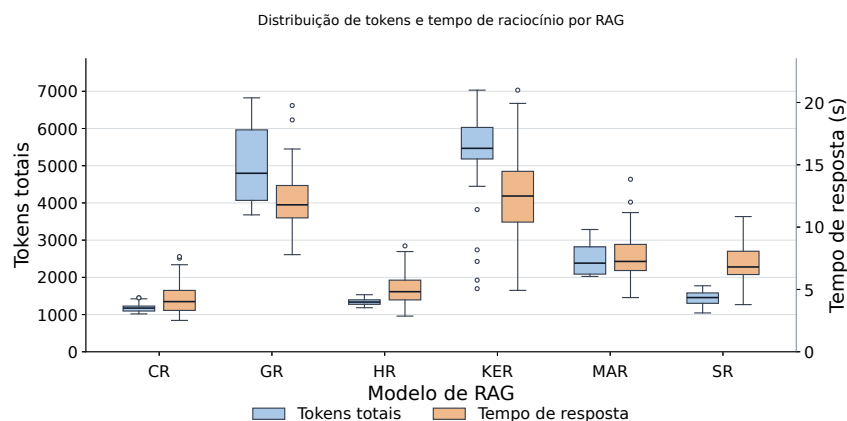


Figura 2. Desempenho dos RAGs com relação ao consumo de tokens e tempo de resposta.

Na Figura 3d referente à métrica de Revocação do contexto, demonstra que todos os modelos, com exceção do *Graph RAG* e *Hybrid RAG*, apresentaram os resultados mais consistentes, com medianas próximas de 1.0 e baixa dispersão, indicando elevada capacidade de recuperar informações relevantes de forma estável. Em contrapartida, o *Graph RAG* e o *Hybrid RAG* exibiram grande dispersão nos resultados (variando entre médias de 0.0 a 1.0 em uma ou mais execuções), evidenciando instabilidade na recuperação contextual.

A Figura 4 apresenta a média das métricas *Faithfulness* e Relevância da resposta por questão para cada abordagem RAG avaliada. As questões foram organizadas em três níveis de dificuldade, conforme descrito na Tabela 2. A partir da análise da Figura 4a,

referente à métrica *Faithfulness*, observa-se que os modelos *Context RAG*, *Hybrid RAG* e *Self RAG* apresentaram os resultados mais consistentes e elevados ao longo das questões (em média entre 0.9 e 1.0). Em especial, esses modelos demonstraram maior estabilidade em questões classificadas como de dificuldade média e alta, mantendo valores próximos de 1.0 em grande parte dos casos. Esse comportamento indica que as respostas geradas permaneceram fortemente alinhadas ao contexto recuperado, reduzindo a ocorrência de alucinações e inferências não fundamentadas.

Em relação à Figura 4b, correspondente à métrica Relevância da resposta, verifica-se que todos os modelos alcançaram valores relativamente elevados (em média entre 0.78 a 0.99), indicando que as respostas produzidas, de maneira geral, mantiveram boa relação semântica com as perguntas realizadas. Entretanto, novamente observa-se destaque para os modelos *Context Rag*, *Hybrid Rag* e *Self Rag*, que apresentaram médias superiores (em média 0.9) e maior estabilidade em praticamente todas as questões avaliadas.

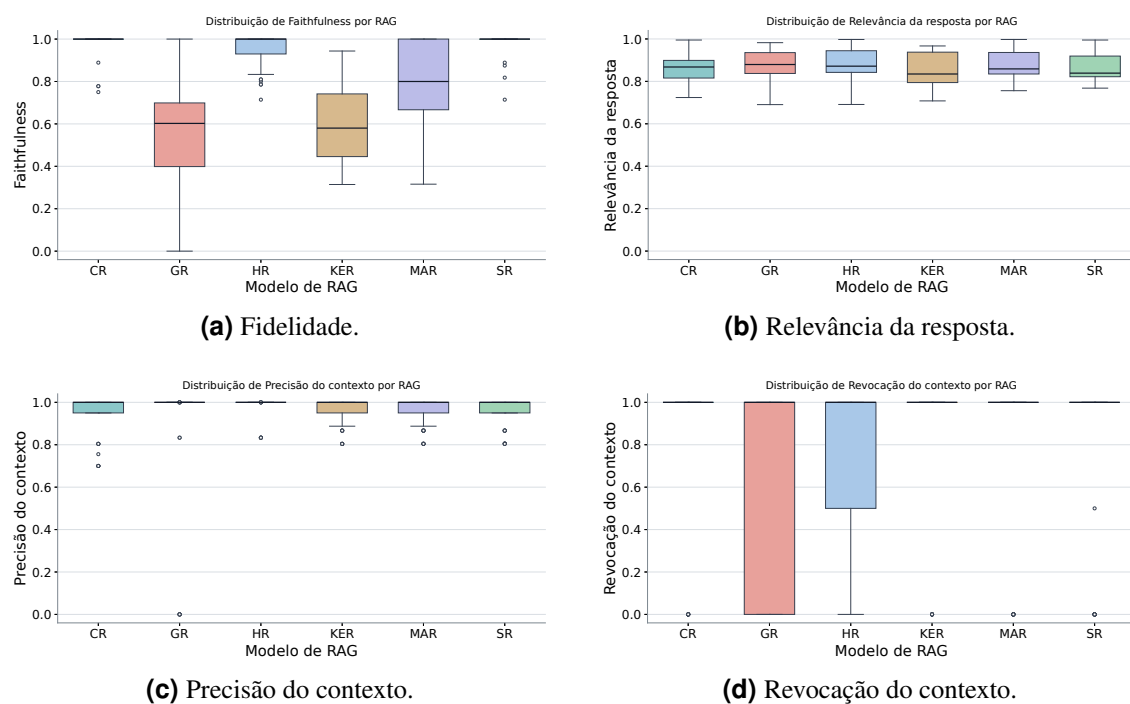
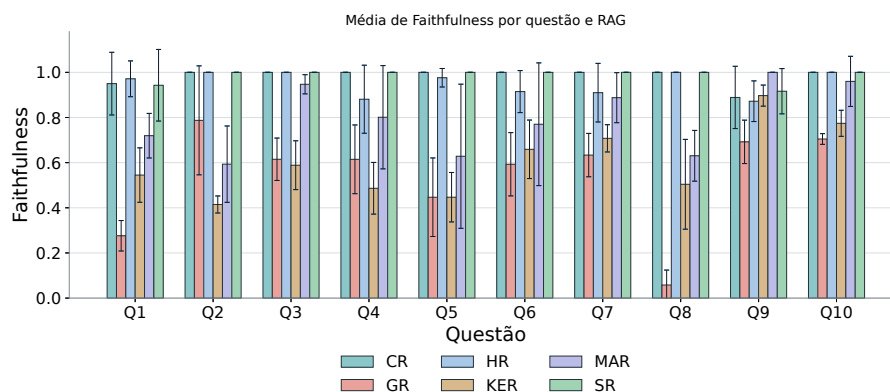
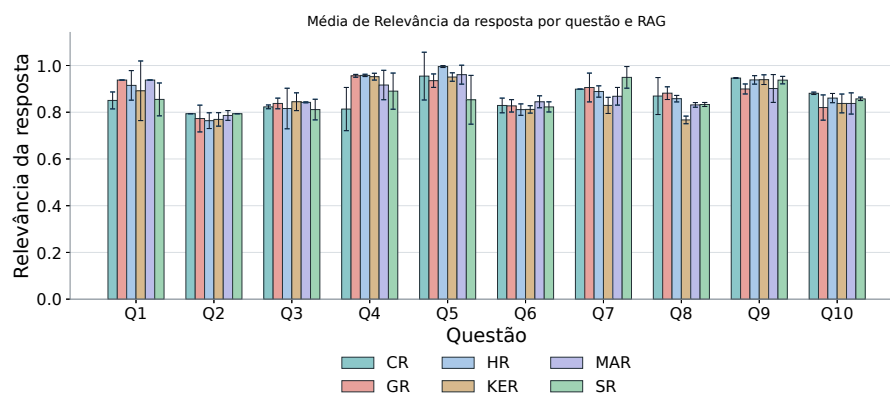


Figura 3. Desempenho dos RAGs para as quatro métricas do framework RAGAS.

Do ponto de vista pedagógico, os resultados técnicos possuem implicações diretas para o ensino: maior *Faithfulness* reduz o risco de o estudante receber explicações incorretas, o que é crítico em disciplinas introdutórias, nas quais conceitos equivocados comprometem etapas subsequentes de aprendizagem; e a menor latência favorece a fluência da interação e o engajamento do aluno. Contudo, as métricas do RAGAS não avaliam diretamente atributos didáticos como clareza da linguagem, adequação ao nível do estudante ou capacidade de oferecer feedback formativo, isto é, guiar o raciocínio do aluno sem simplesmente entregar a resposta pronta. Reconhece-se, assim, que a tradução dos resultados técnicos em benefícios efetivos de aprendizagem depende de validações complementares, como estudos de usabilidade com estudantes reais, sendo esta uma direção prioritária para trabalhos futuros.



(a) Fidelidade por questão e RAG.



(b) Relevância da resposta por questão e RAG.

Figura 4. Desempenho dos RAGs para Fidelidade e Relevância da resposta para cada par pergunta/resposta.

6. Conclusão e Direções Futuras

Este estudo avaliou seis abordagens distintas de Geração Aumentada por Recuperação (RAG) aplicadas ao contexto do ensino de lógica de programação. O objetivo principal foi mitigar as alucinações geradas por modelos de linguagem, garantindo simultaneamente precisão pedagógica e eficiência operacional. Os resultados comprovaram empiricamente que arquiteturas de passo único superaram os modelos mais complexos baseados em grafos. Avaliadas pelas métricas dos frameworks RAGAS e LangSmith, essas soluções garantiram excelente fidelidade factual e relevância semântica, operando com latência reduzida e baixo consumo de tokens.

A principal limitação desta pesquisa reside no uso de um corpus documental restrito a dez materiais e na aplicação de um único modelo de linguagem. Esse escopo limitado tornou as arquiteturas de múltiplas etapas mais robustas que o necessário, impedindo a avaliação sob cenários de alto estresse de recuperação. Como trabalhos futuros, planeja-se expandir significativamente a base de testes para englobar uma gama mais ampla de materiais de outras disciplinas. Adicionalmente, será avaliada a utilização de LLMs de código aberto, visando validar a escalabilidade técnica e a viabilidade financeira em instituições de ensino.

7. Declaração sobre uso de Inteligência Artificial

No presente estudo, foram utilizadas tecnologias de Inteligência Artificial como ferramentas de apoio à revisão da escrita original dos autores, com o objetivo de corrigir erros pontuais e aprimorar a padronização do trabalho.

Referências

- Ahmed, M., Dorrah, M., Ashraf, A., Adel, Y., Elatrozy, A., Mohamed, B. E., and Gomaa, W. (2024). Codeqa: Advanced programming question-answering using llm agent and rag. In *2024 6th Novel Intelligent and Leading Emerging Sciences Conference (NILES)*, pages 494–499. IEEE.
- Alario-Hoyos, C., Kemcha, R., Kloos, C. D., Callejo, P., Estévez-Ayres, I., Santín-Cristóbal, D., Cruz-Argudo, F., and López-Sánchez, J. L. (2024). Tailoring your code companion: Leveraging llms and rag to develop a chatbot to support students in a programming course. In *2024 IEEE International Conference on Teaching, Assessment and Learning for Engineering (TALE)*, pages 1–8. IEEE.
- Asai, A., Wu, Z., Wang, Y., Sil, A., and Hajishirzi, H. (2024). Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *International conference on learning representations*, volume 2024, pages 9112–9141.
- Barzanji, C. and Loitsch, C. (2025). Exploring conversational agents for novice programmers: a scoping review. *Discover Artificial Intelligence*, 5(1):271.
- Borges, J. M. (2025). Desenvolvimento e avaliação de chatbot pedagógico open source com rag para educação online.
- Das, R. and Cheng, Q. (2025). Graphrag-augmented mastery modeling for student programming learning trajectory prediction and explanation via fuzzy clustering. In *2025 IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 3263–3268. IEEE.
- Es, S., James, J., Anke, L. E., and Schockaert, S. (2024). Ragas: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th conference of the european chapter of the association for computational linguistics: system demonstrations*, pages 150–158.
- LangChain, Inc. (2024). Langsmith: The platform for agent engineering. Acesso em: 27 maio 2026.
- Lin, Y., Khan, M. F. F., and Sakamura, K. (2025). Athena: A genai-powered programming tutor based on open-source llm. In *2025 1st International Conference on Consumer Technology (ICCT-Pacific)*, pages 1–4. IEEE.
- Ma, I., Krone-Martins, A., and Videira Lopes, C. (2024). Integrating ai tutors in a programming course. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 1*, pages 130–136.
- Petrin, N. S. M. and Néto, J. C. (2026). Empathiced: um chatbot educacional com rag e computação afetiva em um contexto de iniciação científica coordenada. *ARACÊ*, 8(3):e12683–e12683.

- Prosvirnin, R., Kuznetsov, S., and Jin, S. (2026). Contextrag: Extraction-free hierarchical graph construction for retrieval-augmented generation. *arXiv preprint arXiv:2605.19735*.
- Sarmah, B., Mehta, D., Hall, B., Rao, R., Patel, S., and Pasquali, S. (2024). Hybridrag: Integrating knowledge graphs and vector retrieval augmented generation for efficient information extraction. In *Proceedings of the 5th ACM International Conference on AI in Finance*, pages 608–616.
- Sun, Y., Sun, K., Xu, Y. E., Yang, X., Dong, X. L., Tang, N., and Chen, L. (2025). Kerag: Knowledge-enhanced retrieval-augmented generation for advanced question answering. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 6194–6216.
- Wu, J.-W. and Tseng, M.-H. (2025). Developing a local generative ai teaching assistant system: Utilizing retrieval-augmented generation technology to enhance the campus learning environment. *Electronics*, 14(17):3402.
- Yang, C. C., Liu, G.-J., and Yu, C.-J. (2025). Optimizing rag in programming education: A comparative study of models and strategies. *IEEE Access*, 13:190890–190903.
- Yuan, H., Liu, Z., Qin, M., Qian, H., Shu, Y., Dou, Z., Wen, J.-R., and Sebe, N. (2025). Memory-enhanced retrieval augmentation for long video understanding. *arXiv preprint arXiv:2503.09149*.
- Zhao, H., Wu, Y., Lu, Z., Yu, X., Miao, W., and Chen, L. (2026). Sagejavon: A scalable ai tutor for personalized programming learning. *Information Processing & Management*, 63(5):104605.