Relato de Experiência: Arquitetura de Testes e Validação Semântica para um Sistema RAG no Setor Público

Icaro S. de Oliveira
Universidade Estadual do Ceará (UECE)
Fortaleza, Brasil
icaro.santos@aluno.uece.br

Felipe F. Rodrigues de Oliveira
Companhia de Água e Esgoto do Ceará (CAGECE)
Fortaleza, Brasil
felipe.rodrigues@cagece.com.br

Ismayle S. Santos
Universidade Estadual do Ceará (UECE)
Fortaleza, Brasil
ismayle.santos@uece.br

Resumo

Contexto: Sistemas RAG introduzem não-determinismo que invalida oráculos de igualdade tradicionais. **Método:** Este relato descreve a arquitetura de testes de um assistente RAG para saneamento básico, combinando Playwright, IAM (Identity and Access Management) e validação semântica por cosseno (limiar 0,85) com guardrails determinísticos. **Resultados:** Em 3 meses de CI, 53 casos de teste atingiram 96,22% de aprovação, detectando 2 regressões reais após atualizações regulatórias. **Conclusão:** A arquitetura demonstra que oráculos híbridos (similaridade + conformidade) são viáveis para RAG em ambientes regulatórios, embora a manutenção do golden dataset exija versionamento paralelo à base de conhecimento.

Palavras-chave

Testes Automatizados, Qualidade de Software, Sistemas RAG, Oráculos Semânticos, Integração Contínua

1 Introdução

Diferente de aplicações web convencionais, onde entradas específicas produzem saídas previsíveis, sistemas baseados em Modelos de Linguagem de Grande Porte (LLMs) operam sob uma natureza probabilística. Em arquiteturas de *Retrieval-Augmented Generation* (RAG), o desafio é amplificado: a resposta final depende tanto da qualidade da recuperação de documentos quanto da capacidade de síntese do modelo [5]. Como aponta Adu [1], a integração de IA no teste de software exige novas abordagens para lidar com a fluidez dos resultados gerados. Em contextos industriais, essa necessidade é ainda mais crítica: Min e Budnik [3] demonstram que estratégias tradicionais de verificação e validação são insuficientes para sistemas RAG em produção e propõem critérios específicos para ambientes com restrições operacionais reais.

Este relato apresenta a experiência de automação de testes para um sistema de IA próprio para auxílio de atendimento ao público de uma empresa pública do Ceará. O sistema responde a dúvidas técnicas e regulatórias sobre tratamento e distribuição de água. O foco deste trabalho é a superação do desafio dos oráculos de teste: como validar automaticamente se uma resposta é correta quando ela pode ser escrita de infinitas formas semanticamente equivalentes.

Contribuições. Descrevemos: (i) uma arquitetura em camadas para testes E2E/API e validação semântica; (ii) a aplicação de oráculos híbridos baseados em *embeddings* e *guardrails* determinísticos; e (iii) estratégias de isolamento de estado com IAM (Identity and Access Management) para garantir a repetibilidade em pipelines de Integração Contínua (CI).

2 Trabalhos Relacionados

A avaliação de sistemas RAG tem evoluído para além das métricas clássicas de PLN (Processamento de Linguagem Natural). O framework ARES [5] propõe a avaliação em três dimensões — relevância do contexto, fidelidade da resposta e relevância da resposta em relação à pergunta — e motiva o uso de representações densas (embeddings) para medir fidelidade semântica, em detrimento de métricas de sobreposição de tokens que penalizam paráfrases válidas. Por ser invariante à magnitude dos vetores, a similaridade por cosseno é robusta a respostas de comprimentos variáveis, o que fundamenta sua adoção neste trabalho.

Complementarmente, Wang et al. [6] demonstram que LLMs robustos podem atuar como avaliadores eficazes, atingindo alta concordância com especialistas humanos. Embora essa abordagem (LLM-as-judge) seja promissora, ela introduz dependência de uma chamada de API adicional por caso de teste, o que é inviável em um pipeline de CI com dezenas de execuções diárias e restrições de custo operacional do setor público. Por essa razão, optou-se por um modelo de *embeddings* local (spaCy `pt_core_news_lg`), que executa sem latência de rede e sem custo variável por execução, ainda que com menor capacidade de raciocínio qualitativo. A abordagem de LLM-as-judge permanece como direção para trabalhos futuros, conforme discutido na Seção 6.

Outra abordagem relevante é a geração de exames sintéticos baseados no *corpus* de documentos para medir a acurácia de forma robusta e interpretável [2]. No contexto industrial, Ramachandran [4] enfatiza a necessidade de oráculos sensíveis ao contexto da resposta e recomenda a combinação de verificações semânticas com asserções determinísticas — princípio que orienta diretamente o design do oráculo híbrido descrito na Seção 3.2.

Por fim, Min e Budnik [3] reforçam que, em ambientes industriais com LLMs, a rastreabilidade e o isolamento de estado entre execuções são requisitos não negociáveis para a validade dos resultados — motivação direta para o uso do IAM com *global setup*, descrito na Seção 3.1.

3 Arquitetura de Testes Proposta

A infraestrutura foi organizada em quatro fases sequenciais, apresentadas na Figura 1. Das cinco suítes, apenas a de validação semântica é específica de RAG; as suítes de API, performance, setup e teardown constituem infraestrutura convencional, reaproveitada para garantir isolamento de estado e rastreabilidade — requisitos que o não-determinismo do LLM torna mais críticos, mas não altera tecnicamente. A suíte opera em dois modos: simulação, com validação por vetores locais para execuções rápidas de regressão,

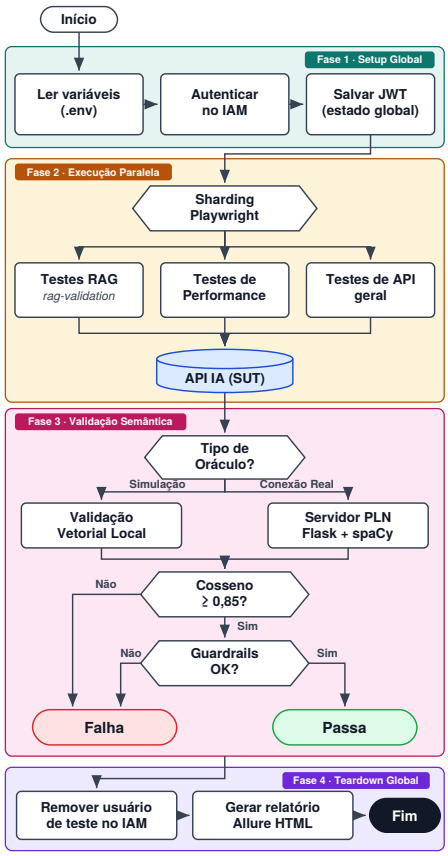

Figura 1: Arquitetura de testes em 4 fases

e conexão real, que consulta o servidor de PLN e o sistema em produção.

3.1 Camada de Execução e Orquestração

Utilizamos o **Playwright** pelo suporte nativo a aplicações reativas e a *sharding*: a suíte é particionada em múltiplos nós para reduzir o tempo de *feedback*, fator crítico dado que inferências de LLM são computacionalmente lentas. Os testes são versionados no Azure DevOps, que dispara a execução no Jenkins[1], responsável por orquestrar o ciclo completo. Os resultados são consolidados no Allure Report[2], que agrega histórico de execuções e permite inspecionar cada caso individualmente — recurso explorado na Seção 4.1 para diagnosticar falhas limítrofes. Sistemas do setor público exigem autenticação rigorosa; por isso, um *global setup* autentica-se no **IAM** via API antes da suíte e salva o *storage state*, evitando que cada caso repita o fluxo de login na interface e reduzindo tempo de execução e fragilidade (*flakiness*).

[1]https://www.jenkins.io
[2]https://allurereport.org

3.2 Validação Semântica e Oráculos

O coração da arquitetura é a camada de validação semântica, na qual, seguindo princípios de avaliação de RAG [5, 6], implementamos dois oráculos. O primeiro, de **Similaridade Semântica**, usa o modelo do spaCy para converter a resposta do sistema e a "resposta padrão" (*golden answer*) em vetores densos, calculando a similaridade por cosseno com limiar de 0,85 para aprovação automática. O segundo, de **Restrições Negativas (*Guardrails*)**, aplica verificações determinísticas de *must-contain* e *must-not-contain*, já que a similaridade pode ignorar detalhes factuais críticos (ex.: inverter um "sim" por "não"). Os oráculos são conjuntivos: um caso só é aprovado se a similaridade atingir o limiar *e* todas as restrições forem satisfeitas — qualquer violação de *guardrail* reprova o caso independentemente do *score*. A similaridade contra a *golden answer* atua como *proxy* de correção (*correctness*) da resposta; relevância e fidelidade ao contexto recuperado não são medidas diretamente, sendo parcialmente cobertas pelos *guardrails*.

4 Resultados e Discussão

A implementação da arquitetura proposta foi validada em um ambiente de integração contínua real, com execuções registradas ao longo de aproximadamente três meses de desenvolvimento ativo do sistema RAG. O pipeline completo, orquestrado via Jenkins, executa em cerca de **25 minutos** por ciclo, sendo 19 minutos e 22 segundos destinados à inicialização do servidor PLN e 4 minutos e 58 segundos à execução da suíte Playwright. Essa distribuição revela que o gargalo de tempo está no *build* do componente de validação semântica, não nos testes em si — informação relevante para equipes que desejam otimizar o *feedback loop*.

Na execução mais recente registrada (23/04/2026), a suíte continha **53 casos de teste**, distribuídos em cinco suítes: testes semânticos (21), testes de API (18), testes de performance (12), *setup* (1) e *teardown* (1). A taxa de aprovação atingiu **96,22%**, com apenas **2 falhas categorizadas como *product defects*** — ou seja, defeitos reais do sistema e não ruído do oráculo. Esses resultados, consolidados no Allure Report gerado automaticamente a cada execução, demonstram que a arquitetura é capaz de distinguir regressões genuínas de instabilidades do ambiente.

A implementação permitiu identificar regressões que testes manuais dificilmente detectariam, notadamente a degradação das respostas após atualizações na base de conhecimento (PDFs de regulamentação sobre tratamento e distribuição de água). O gráfico de tendência do Allure evidencia um pico de falhas em torno da execução #31, correlacionado com uma atualização de documentos regulatórios, seguido de estabilização após a correção do *prompt* de sistema e reindexação do corpus.

Calibração do Limiar Semântico. O limiar de similaridade por cosseno de **0,85** foi determinado empiricamente por meio de uma análise de sensibilidade sobre o *golden dataset*. Para cada par pergunta–resposta do conjunto, calculamos a similaridade entre a *golden answer* e variações manuais criadas pela equipe de QA em três categorias: (i) respostas semanticamente equivalentes com paráfrase; (ii) respostas com erro factual sutil (ex.: inversão de valores numéricos); e (iii) respostas completamente incorretas. Observamos que respostas equivalentes concentravam-se acima de 0,87, enquanto respostas com erro factual sutil raramente ultrapassavam 0,83. O

valor 0,85 foi escolhido como ponto de equilíbrio que minimiza simultaneamente falsos positivos (aceitar respostas erradas) e falsos negativos (rejeitar paráfrases válidas). Casos limítrofes — como o *score* de 0,84 discutido na Seção 4.1 — são sinalizados para revisão manual pelo analista de QA antes de qualquer ajuste do limiar.

4.1 Lições Aprendidas

A manutenção do *golden dataset* mostrou-se o maior desafio operacional da arquitetura. O conjunto é composto por pares pergunta–*golden answer* curados manualmente por especialistas do domínio de saneamento básico, versionados no repositório junto ao código de testes. A seleção das perguntas segue dois critérios: cobertura dos fluxos críticos de atendimento e representatividade dos documentos regulatórios mais consultados. Toda alteração na base de conhecimento (novo PDF ou revisão de norma) dispara uma revisão obrigatória das *golden answers* afetadas, garantindo que o "exame" evolua junto com o sistema [2].

Descobrimos que falhas limítrofes (ex.: *score* 0,84) exigem uma camada de observabilidade que permita ao analista de QA revisar a falha rapidamente e decidir se o limiar deve ser ajustado ou se o modelo de fato alucinou. Para isso, os relatórios Allure são configurados para exibir o *score* exato de cada teste semântico reprovado, acelerando o diagnóstico.

5 Ameaças à Validade

Validade interna. A principal ameaça é o uso de um modelo de *embeddings* de propósito geral (spaCy), que pode não capturar com precisão as nuances técnicas do domínio de saneamento. Além disso, o limiar semântico de 0,85 foi calibrado com o *golden dataset* atual, podendo exigir reavaliações periódicas conforme a base de dados evolui.

Validade externa. Os resultados baseiam-se em um único sistema RAG do setor de saneamento brasileiro, utilizando normas técnicas em português. Embora a arquitetura em camadas e o uso do Playwright com IAM sejam independentes de domínio e transferíveis, a generalização de componentes específicos (como o limiar semântico e os *guardrails*) para outros idiomas, setores ou LLMs demanda validação adicional.

Validade de construto. A similaridade por cosseno avalia a proximidade vetorial, não a correção factual direta. Embora *guardrails* determinísticos (*must-contain* e *must-not-contain*) mitiguem parcialmente esse risco para conceitos críticos, sua cobertura não é exaustiva. Trabalhos futuros empregando *LLM-as-judge* [6] podem aprimorar essa validação factual.

Validade de conclusão. A latência variável da API de IA pode gerar *timeouts* intermitentes no CI, exigindo políticas de *retry* que podem mascarar falhas reais. Ademais, a alta taxa de aprovação (96,22%) reflete a atual maturidade do sistema, contrastando com a maior variância e picos de falhas (ex.: ciclo #31) observados nas fases iniciais de desenvolvimento.

6 Conclusão

Este relato demonstrou que a validação de sistemas RAG em contextos industriais e governamentais exige uma transição de oráculos de igualdade para oráculos de similaridade e conformidade. A arquitetura proposta, integrando Playwright e validação semântica, provou-se eficaz para garantir que o assistente de atendimentos desse setor público forneça informações confiáveis aos seus usuários. Trabalhos futuros focarão em LLM-as-judge [6] para avaliações qualitativas mais profundas — hoje inviável no caminho crítico do CI pelo custo por chamada, mas plausível via inferência local com modelos quantizados ou execução assíncrona em ciclos noturnos, fora do feedback loop de commit.

Disponibilidade de Artefatos

O golden dataset e o corpus regulatório não podem ser disponibilizados por conterem informações operacionais internas da organização; todo o material aqui apresentado passou por revisão de anonimização conduzida pela coordenação de segurança da informação. Uma implementação de referência aberta da arquitetura de testes, licenciada sob Apache 2.0, está disponível em https://doi.org/10.5281/zenodo.22117826, com espelho em https://github.com/Icar0S/Evaluating-Answer-Quality-RAG-System.

Referências

[1] Gilbert Adu. 2024. *Artificial Intelligence in Software Testing: Test scenario and case generation with an AI model (gpt-3.5-turbo) using Prompt engineering, Fine-tuning and Retrieval augmented generation techniques.* Master's thesis. Itä-Suomen yliopisto.

[2] Gauthier Guinet, Behrooz Omidvar-Tehrani, Anoop Deoras, and Laurent Callot. 2024. Automated evaluation of retrieval-augmented language models with task-specific exam generation. *arXiv* abs/2405.13622 (2024), 1–20.

[3] Ziran Min and Christof J Budnik. 2025. Verification and validation of llm-rag for industrial automation. In *2025 IEEE International Conference on Artificial Intelligence Testing (AITest)*. IEEE, IEEE, 50–53.

[4] Sooraj Ramachandran. 2025. Evaluating AI Responses: A Step-by-Step Approach for Test Automation. *The Eastasouth Journal of Information System and Computer Science* 2, 03 (2025), 381–390.

[5] Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. Ares: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 338–354.

[6] Yang Wang, Alberto Garcia Hernandez, Roman Kyslyi, and Nicholas Kersting. 2024. Evaluating quality of answers for retrieval-augmented generation: A strong llm is all you need. *arXiv* abs/2406.18064 (2024), 1–20.