Medição da Qualidade de Recuperação e Geração em LLMs com o Framework RAGAS

Gustavo S. Vasconcelos, João Paulo de S. Wakugawa, Bruno M. Nogueira (Orientador)

[1]Faculdade de Computação – Fundação Universidade Federal de Mato Grosso do Sul (UFMS)
Campo Grande – MS – Brasil

{gustavo_vasconcelos, paulo.wakugawa, bruno.nogueira}@ufms.br

Resumo. *Este trabalho apresenta uma análise comparativa de Grandes Modelos de Linguagem (LLMs) aplicados a um agente conversacional que utiliza a técnica de Geração Aumentada por Recuperação (RAG). O estudo foi realizado na plataforma Inithub, utilizando dados reais de estudantes e dados sintéticos gerados por automação para simular diferentes cenários de uso. Foram avaliados três modelos de última geração: OpenAI GPT-5-Chat, Google Gemini 2.5 Pro e Anthropic Claude 3.5 Sonnet. A validação utilizou o framework Ragas, focando nas métricas de Fidelidade (Faithfulness) e Relevância (Answer Relevancy). Ao final, são discutidos os diferentes comportamentos observados em cada modelo e como essas características influenciam a construção de assistentes virtuais para gestão de inovação.*

Abstract. *This work presents a comparative analysis of Large Language Models (LLMs) applied to a conversational agent utilizing Retrieval-Augmented Generation (RAG). The study was conducted on the Inithub platform, using real data from students and synthetic data generated via automation to simulate different usage scenarios. Three state-of-the-art models were evaluated: OpenAI GPT-5-Chat, Google Gemini 2.5 Pro, and Anthropic Claude 3.5 Sonnet. The validation used the Ragas framework, focusing on Faithfulness and Answer Relevancy metrics. Finally, the work discusses the different behaviors observed in each model and how these characteristics influence the development of virtual assistants for innovation management.*

1. Introdução

A evolução crescente dos Grandes Modelos de Linguagem (Large Language Models - LLMs) transformou o cenário tecnológico global, impulsionando a adoção acelerada de Inteligência Artificial em diversos setores [McKinsey & Company 2024]. No entanto, [Zhao et al. 2023], apesar da capacidade de gerar textos coerentes, esses modelos ainda possuem limitações intrínsecas, principalmente quando aplicados em tarefas que exigem conhecimentos específicos. Por serem treinados em grandes volumes de dados públicos, quando colocados em situações em que contextos privados e regras de negócios são exigidos, acabam gerando respostas imprecisas ou não fundamentadas [Brown et al. 2020, Ji et al. 2023].

Nesse contexto, surgiu a necessidade de ter mecanismos que permitissem às LLMs acessar informações externas. Dessa forma, a arquitetura RAG (Retrieval-Augmented Generation) vem sendo cada vez mais adotada pela indústria. Conforme [Lewis et al. 2020],

nessa abordagem, um mecanismo de recuperação busca documentos relevantes em uma base externa e os fornece como contexto para a LLM, aumentando a precisão e a fundamentação das respostas. Entretanto, conforme [Simon et al. 2024], o RAG não é uma solução única ou monolítica, mas um sistema composto por diversas escolhas de design, como o tipo de modelo gerador e o método de recuperação, o que torna seu desenvolvimento altamente experimental. Sendo assim, torna-se essencial medir o impacto real dessas decisões para garantir que o sistema seja confiável e não dependa apenas de casos isolados ou situações específicas.

A motivação do presente trabalho surge da necessidade de mensurar, de forma prática, como a escolha do Modelo de Linguagem influencia a eficácia global de sistemas RAG, impactando desde a interpretação da intenção do usuário até a geração da resposta final. Para trazer essa análise para a realidade, utilizamos o Inithub, uma ferramenta desenvolvida durante uma competição de inovação tecnológica acadêmica (Pantanal.dev). O Inithub funciona como um repositório de 'iniciativas', propostas estratégicas para resolver problemas ou gerar valor organizacional. A plataforma conta com um agente conversacional que auxilia no cadastro e utiliza RAG para consultar o banco de dados continuamente. Assim, quando o sistema identifica um contexto similar, ele apresenta a iniciativa encontrada ao usuário, deixando em suas mãos a decisão: seguir com uma nova ideia ou colaborar com o projeto que já existe. Além disso, três LLMs diferentes foram utilizadas no experimento, variando de acordo com a sessão do usuário: OpenAI GPT-5-Chat, Google Gemini 2.5 Pro e Anthropic Claude 3.5 Sonnet. Dessa forma, podemos ver como as diferentes capacidades de raciocínio e síntese de cada modelo determinam a qualidade da experiência entregue ao usuário.

A validação do sistema foi feita usando o framework Ragas [Shahul Es 2023] devido a sua capacidade de ser reference-free, isto é, elimina a necessidade de possuir um conjunto de respostas ideais humanas (ground truth), algo que raramente está disponível no desenvolvimento ágil de software. A análise foca em duas métricas principais: Faithfulness, que garante que a resposta seja estritamente derivada do contexto para prevenir alucinações; e Answer Relevancy, que penaliza respostas incompletas ou redundantes.

Este trabalho está organizado da seguinte maneira: O Capítulo 2 apresenta a fundamentação teórica sobre LLMs, a arquitetura RAG e o framework de avaliação Ragas. O Capítulo 3 detalha a metodologia, descrevendo a implementação do agente no Inithub, a arquitetura agêntica e o desenho do experimento. O Capítulo 4 apresenta os resultados quantitativos e qualitativos obtidos. O Capítulo 5 discute esses achados, correlacionando o desempenho observado com as características de cada modelo. Por fim, o Capítulo 6 apresenta a conclusão, limitações do estudo e sugestões para trabalhos futuros.

2. Fundamentação Teórica

2.1. Grandes Modelos de Linguagem

No campo da inteligência artificial generativa, os Grandes Modelos de Linguagem (LLMs) representam um avanço significativo. Possuem a função de processar e gerar texto de maneira coerente, simulando a forma de escrita e raciocínio humano. Baseados na arquitetura *Transformers*, proposta por [Vaswani et al. 2017], esses modelos se diferenciam de abordagens anteriores por utilizarem um mecanismo de autoatenção (*Self-Attention*).

Modelos antigos como RNN ou LSTM processam textos de forma sequencial, o que tornava o treinamento lento e impunha uma grande limitação no que se refere a manter contextos longos na memória [Bengio et al. 1994, Hochreiter and Schmidhuber 1997]. O mecanismo de atenção, por outro lado, permite que a LLM analise todo o contexto fornecido de uma única vez, mapeando quais palavras são mais relevantes para responder a uma determinada pergunta, mesmo que a distância entre elas seja grande. Isso é crucial para aplicações RAG, pois permite que o modelo receba grandes volumes de documentos como contexto e consiga encontrar a informação correta para gerar a resposta [Lewis et al. 2020, Gao et al. 2023].

2.2. Natureza Generativa e Limitações

Apesar da alta capacidade de compreensão, os grandes modelos de linguagem são probabilísticos, treinados para prever a próxima palavra mais provável em uma sequência [Brown et al. 2020]. Logo, as LLMs não ”sabem”de fato a informação; elas apenas reproduzem padrões linguísticos aprendidos. Isso acarreta duas limitações principais citadas por [Zhao et al. 2023], as quais motivam este trabalho:

1. **Conhecimento estático:** Os modelos estão limitados ao que aprenderam no momento do treinamento, não sendo possível solicitar informações posteriores ao corte de conhecimento (*cutoff*).
2. **Alucinação:** Por não ter a informação clara ou acesso a fatos, o modelo tende a inventar informações para completar o padrão probabilístico de forma plausível.

2.3. Geração Aumentada por Recuperação (RAG)

Tendo em vista as limitações supracitadas, [Lewis et al. 2020] introduziram o conceito de RAG. O RAG une o conhecimento pré-treinado (memória paramétrica) com uma memória proveniente de um banco de dados externo contendo informações atualizadas sobre determinado contexto (não paramétrica).

Essa arquitetura híbrida funciona em duas etapas. O mecanismo de recuperação (*Retriever*) localiza as informações mais relevantes em uma base de conhecimento externa conforme a entrada do usuário. Após isso, o mecanismo gerador (*Generator*) recebe os documentos concatenados ao *prompt* original e os utiliza como contexto para formular uma resposta fundamentada. Essa abordagem permite utilizar o sistema em contextos específicos simplesmente adicionando novos arquivos à base, eliminando a necessidade de retreinar a rede neural, o que é inviável para a maioria das organizações [Gao et al. 2023].

2.4. Recuperação Vetorial e Embeddings

Para recuperar informações baseadas no significado e não apenas em palavras-chave exatas, utilizam-se os *Embeddings*. Segundo [Reimers and Gurevych 2019], na proposta do Sentence-BERT, o objetivo é derivar representações vetoriais onde frases semanticamente similares estejam próximas no espaço vetorial.

No processo de indexação, os documentos textuais são convertidos em vetores numéricos de tamanho fixo. Quando é feita uma consulta, ela também é vetorizada, tornando possível calcular a similaridade semântica através de operações matemáticas, como a distância de Cosseno. Essa abordagem diminui o custo computacional em comparação ao uso de modelos BERT puros, viabilizando a recuperação em tempo real.

2.5. RAG Modular e Agêntico

O RAG original de [Lewis et al. 2020] é descrito como um fluxo linear (Buscar $\rightarrow$ Responder), mas essa arquitetura evoluiu para formas mais complexas de implementação. Conforme destacam [Gao et al. 2023], as limitações da abordagem padrão (*Naive RAG*) impulsionaram a evolução para o que é conhecido hoje como *Modular RAG*.

Diferente do modelo linear, o RAG modular possui uma arquitetura composta por módulos funcionais independentes, o que a torna mais flexível. Segundo o levantamento, isso permite a utilização de estratégias mais avançadas, como o *Routing* (roteamento) e a recuperação adaptativa. Nesse cenário, a LLM passa a atuar na orquestração do sistema, possuindo autonomia para avaliar a necessidade de busca, reescrever consultas para melhor alinhamento ou selecionar diferentes fontes de dados, adequando-se à complexidade da entrada do usuário.

2.6. Métricas de Avaliação e Framework RAGAS

A avaliação de sistemas de geração de texto, como agentes conversacionais, apresenta desafios significativos. Métricas baseadas em n-gramas, como BLEU e ROUGE, calculam a qualidade verificando quantas palavras a resposta da máquina tem em comum com uma resposta humana ideal. Porém, [Liu et al. 2016] demonstraram que essas métricas baseadas na contagem de palavras não funcionam bem em sistemas de diálogos. De acordo com o estudo, isso ocorre porque, em conversas reais, as respostas corretas possuem diversas variações que não usam necessariamente as mesmas palavras.

Para superar essa limitação, o framework RAGAS (*Retrieval Augmented Generation Assessment*) propõe o paradigma de "LLM como Juiz"(*LLM-as-a-Judge*). Nessa abordagem, utiliza-se um modelo de linguagem avançado para avaliar semanticamente a tripla composta por: Pergunta (q), Contexto Recuperado ($c(q)$) e Resposta Gerada ($a_s(q)$). Conforme [Shahul Es 2023], isso permite uma avaliação sem a necessidade de respostas humanas pré-anotadas.

O framework Ragas disponibiliza um conjunto abrangente de métricas para avaliar diferentes componentes do pipeline RAG, divididas em duas categorias: métricas baseadas em referência (que exigem um gabarito humano ou *ground truth*) e métricas livres de referência (*reference-free*).

Dada a proposta deste trabalho de avaliar o sistema em um cenário de desenvolvimento ágil, onde a anotação manual de gabaritos é custosa ou inexistente, selecionaram-se para este estudo apenas as métricas capazes de operar exclusivamente com a tripla (Pergunta, Contexto, Resposta). A seguir, detalham-se as definições formais das duas métricas escolhidas que atendem a este critério.

2.6.1. Faithfulness

Essa métrica avalia a consistência real da resposta gerada em relação ao contexto recuperado e serve como principal indicador de alucinações. O processo de cálculo descrito pelos autores ocorre em duas etapas:

1. **Decomposição:** Uma LLM é utilizada para extrair um conjunto de declarações

atômicas $S(a_s(q))$ a partir da resposta gerada, decompondo sentenças longas em afirmações curtas e focadas.
2. **Verificação:** Para cada declaração s_i, o modelo verifica se ela pode ser inferida logicamente a partir do contexto $c(q)$.

A pontuação de fidelidade (F) é calculada pela razão entre o número de declarações suportadas pelo contexto ($|V|$) e o número total de declarações extraídas ($|S|$):

$$F = \frac{|V|}{|S|} \tag{1}$$

2.6.2. Answer Relevancy

Essa métrica mensura o quão pertinente é a geração em relação à pergunta original e penaliza respostas incompletas ou redundantes. O cálculo não avalia a veracidade, mas a diretividade.

O algoritmo utiliza uma abordagem de engenharia reversa: solicita-se à LLM que gere n perguntas artificiais (q_i) baseadas exclusivamente na resposta gerada $a_s(q)$. Em seguida, calcula-se a similaridade de cosseno (sim) entre os embeddings dessas perguntas geradas e o embedding da pergunta original q. A pontuação final (AR) é a média dessas similaridades:

$$AR = \frac{1}{n} \sum_{i=1}^{n} sim(q, q_i) \tag{2}$$

Uma pontuação alta indica que a resposta gerada é tão específica que permite reconstruir a pergunta original com alta precisão.

No presente trabalho, estes conceitos e métricas foram aplicados para validar o agente conversacional da plataforma Inithub, submetendo modelos de diferentes famílias (GPT, Gemini e Claude) ao mesmo fluxo de avaliação. O processo de implementação do agente, a arquitetura agêntica baseada em grafos e o desenho do experimento são detalhados na próxima seção.

3. Metodologia

Neste capítulo, é feita a descrição dos procedimentos adotados para o desenvolvimento do agente conversacional, a estratégia de coleta de dados e o desenho experimental para avaliar a qualidade das respostas geradas pelas LLMs. A abordagem do experimento se divide em três etapas principais: (i) implementação da arquitetura agêntica na plataforma Inithub; (ii) construção do dataset contendo interações reais e sintéticas; e (iii) aplicação do pipeline de avaliação automática utilizando o framework Ragas.

3.1. Ambiente Experimental: Plataforma Inithub e Arquitetura Agêntica

Para realizar as validações, foi utilizado o *Inithub*, uma plataforma de iniciativas de inovação desenvolvida durante a competição acadêmica Pantanal.dev. O sistema funciona via interface web e se comunica com o servidor através de conexões baseadas em WebSocket, implementadas via framework FastAPI.

O agente foi construído utilizando a biblioteca `LangGraph`, que permite modelar o fluxo de conversação como um Grafo de Estados (*StateGraph*). Essa abordagem fornece ao sistema a capacidade de orquestração, permitindo que o agente decida dinamicamente qual ferramenta ou fluxo seguir com base na entrada do usuário.

O estado da aplicação (*State*) é compartilhado entre diferentes nós de processamento. A estrutura do estado foi definida via Pydantic e armazena o histórico de mensagens, os dados da mensagem, os dados da iniciativa (título, contexto, tema) e as iniciativas similares recuperadas.

O fluxo de decisão e roteamento do agente é descrito formalmente pelo Algoritmo 1, que abstrai a lógica implementada:

Algoritmo 1 Lógica de Decisão e Roteamento do Agente

Entrada: Mensagem do usuário (M), Histórico da Sessão (H)
Saída: Resposta gerada (R)

$Estado \leftarrow AtualizarContexto(H, M)$
$Intencao \leftarrow ClassificarIntencao(Estado)$ ▷ Usa LLM para classificar fluxo
$Dados \leftarrow ExtrairEntidades(Estado)$ ▷ Identifica título, tema, etc.
if $Intencao$ == "Consultar Iniciativa" **then**
    $Contexto \leftarrow BuscarVetorial(Dados)$ ▷ Recuperação RAG
    $R \leftarrow GerarResposta(Contexto, Dados)$
else if $Intencao$ == "Registrar Iniciativa" **then**
    $Similar \leftarrow BuscarVetorial(Dados)$
    $R \leftarrow GuiarCadastro(Dados, Similar)$
else ▷ Fluxo de Direcionamento/Dúvidas Gerais
    $R \leftarrow ResponderConversa(Estado)$
end if
return R

Conforme detalhado no algoritmo, o processamento ocorre nas seguintes etapas lógicas:

1. **Classificação de Intenção:** A entrada do usuário é submetida a uma LLM configurada para saída estruturada, classificando a intenção em três categorias: *direcionar*, *consultar* ou *registrar*.
2. **Extração de Entidades:** Independentemente da intenção, o sistema tenta extrair informações relevantes da mensagem (como tema ou entregáveis) para preencher o objeto de domínio `Initiative`.
3. **Roteamento Condicional:** Com base na classificação, o grafo direciona o fluxo para um dos nós especialistas:
    - **Guide:** Para dúvidas gerais e orientação.
    - **Find Initiative:** Executa o pipeline RAG para buscar iniciativas similares existentes.
    - **Register Initiative:** Auxilia o usuário no preenchimento dos campos obrigatórios para cadastro.

Dessa forma, isolamos a lógica de recuperação no nó de consulta, fazendo com que seja mais fácil substituir e comparar os LLMs sem afetar o restante do sistema.

3.2. Aquisição e Geração de Dados

Visando possibilitar uma avaliação relevante, construiu-se um conjunto de dados combinando interações reais de usuários com dados sintéticos gerados por automação.

3.2.1. Coleta de Interações Reais

Para a coleta de dados reais, a plataforma foi disponibilizada para um grupo variado de pessoas (versão demo). Para capturar as interações sem interferir na performance da aplicação, desenvolveu-se um mecanismo de interceptação baseado no padrão de projeto *Decorator*.

O decorador é aplicado aos nós de geração de resposta e atua como um middleware de observabilidade. A cada execução bem-sucedida, o decorador captura assincronamente:

- O prompt do usuário e a resposta gerada pelo agente;
- Os metadados do modelo utilizado (nome e versão);
- O contexto recuperado da base vetorial (RAG);
- O identificador da sessão.

Esses dados foram enviados automaticamente para uma planilha externa (Google Sheets), persistindo as triplas (pergunta, contexto, resposta) necessárias para a avaliação *reference-free*. Nesta etapa, registrou-se a participação de 26 usuários distintos.

3.2.2. Geração de Dados Sintéticos e Automação

Tendo em vista o baixo volume de amostras reais, foi empregada uma estratégia de aumento de dados, simulando usuários. Para isso, foi desenvolvido um script de automação utilizando a biblioteca (*Selenium*) para interagir diretamente com a interface web do Inithub.

A automação baseia-se em um agente autônomo, alimentado pelo modelo `gpt-4o-mini`, instruído a simular o fluxo de cadastro de iniciativas. O prompt do agente definiu personas com diferentes níveis de clareza e objetividade, gerando variações linguísticas e cenários de testes diversificados.

Visando a coerência das simulações, o agente foi alimentado com exemplos de cenários. A Figura 1 exemplifica a estrutura das iniciativas, utilizada como base para que o agente formulasse suas mensagens de cadastro.

Figura 1. Exemplo de cenário semente (*Seed Scenario*) utilizado na automação.

```
{
    "title": "Sistema de Gestão de Biblioteca Digital",
    "theme": "Tecnologia e Educação",
    "context": "A biblioteca usa sistema manual...",
    "description": "Plataforma web para reservas...",
    "deliverable": "Sistema web completo...",
    "evaluation_criteria": "Redução do tempo em 70%."
}
```

Fonte: Elaborado pelo autor.

Com isso, acrescentamos 252 mensagens ao dataset, ampliando a base de testes para a análise de robustez e consistência das métricas.

3.3. Pipeline de Recuperação e Geração

O componente central do experimento é o módulo RAG, acionado no nó de busca de iniciativa. O processo de recuperação utiliza busca semântica vetorial. As iniciativas cadastradas no banco de dados foram convertidas em embeddings utilizando o modelo `text-embedding-3-small`.

Durante a execução, a consulta do usuário é vetorizada e comparada com o banco vetorial através da similaridade do cosseno. Os documentos com a maior pontuação de similaridade são injetados no prompt do sistema como contexto.

A implementação desta lógica pode ser observada no Código 2, que ilustra como o nó de busca orquestra a recuperação dos embeddings e a construção do prompt final enviado ao modelo.

Figura 2. Implementação do nó de recuperação e geração (RAG).

```
@decorators.log_node
@decorators.with_prompt()
@decorators.send_test_case()
def find_initiative_v1(state: State, prompt_template=None):
    initiative = state.get("initiative")

    # 1. Etapa de Recuperação (Retrieval)
    similar_initiatives = (
        backend.find_similar_embeddings(initiative) if initiative else []
    )

    # 2. Injeção de Contexto (Augmentation)
    prompt_content = (prompt_template or "").format(
        SIMILAR_INITIATIVES=similar_initiatives
    )

    # 3. Geração da Resposta (Generation)
    result = {
        "messages": default_llm.invoke(
            state["messages"]
            + [
                {
                    "role": "system",
                    "content": prompt_content,
                }
            ],
        ),
        "similar_initiatives": similar_initiatives,
    }
    return result
```

Fonte: Elaborado pelo autor.

Para avaliar o impacto do modelo gerador, o sistema foi configurado para alternar dinamicamente entre três LLMs de última geração, acessadas via API **OpenRouter**, um gateway que unifica o acesso a diferentes famílias de modelos.

- **OpenAI GPT-5-Chat** [OpenAI 2025]: Selecionado por representar um ”sistema unificado”que integra capacidades de resposta rápida com um módulo de reflexão estendida (*thinking*). A documentação de lançamento destaca uma redução significativa na taxa de alucinações (até 80% menos erros factuais com reflexão ativa) e um aprimoramento no cumprimento de instruções complexas, fatores críticos para garantir a fidelidade das respostas no contexto do RAG.
- **Google Gemini 2.5 Pro** [Gemini Team, Google 2025]: Selecionado por sua nova capacidade de raciocínio estendido (”thinking model”), projetada para lidar com tarefas complexas antes de gerar uma resposta. Além disso, seu suporte a contextos longos foi um fator determinante, permitindo que o sistema analise um grande volume de iniciativas recuperadas sem perder informações relevantes.
- **Anthropic Claude 3.5 Sonnet** [Anthropic 2024]: Escolhido por sua capacidade aprimorada de compreender nuances e seguir instruções complexas. O modelo destaca-se na geração de textos com tom natural e ”relacionável”, além de demonstrar alta proficiência em orquestrar fluxos de trabalho de múltiplas etapas (*multi-step workflows*), o que é crucial para a tomada de decisão dentro da arquitetura agêntica do sistema.

3.4. Avaliação Automatizada com Ragas

Dada a inviabilidade de avaliar manualmente centenas de interações, utilizou-se o framework *Ragas* (versão $\geq$ 0.2.15) para computar as métricas de qualidade de forma automática e em lote, processando o arquivo CSV consolidado.

Para a execução deste pipeline de avaliação, instanciou-se o modelo GPT-4o como ”LLM-Juiz”, responsável por calcular os scores das métricas selecionadas, conforme detalhado na fundamentação teórica:

1. **Faithfulness:** Aplicada para verificar a aderência estrita da resposta aos `retrieved_contexts` capturados pelo decorador, penalizando alucinações extrínsecas.
2. **Answer Relevancy:** Aplicada para mensurar a objetividade da resposta em relação ao *prompt* original do usuário.

Os escores resultantes, normalizados no intervalo $[0, 1]$, foram agregados por modelo para compor as tabelas de análise apresentadas no capítulo de resultados.

3.5. Reprodutibilidade e Ética

Visando garantir a transparência e a reprodutibilidade deste estudo, os artefatos de software e os procedimentos de avaliação foram disponibilizados publicamente.

- **Código Fonte do Sistema:** A implementação completa do agente e da plataforma Inithub encontra-se no repositório: `https://github.com/engsoft-pantanal-dev/inithub/tree/main`.

- **Pipeline de Avaliação:** O notebook contendo a configuração do Ragas, carregamento dos datasets e o cálculo das métricas está disponível em: https://colab.research.google.com/drive/1LD9yEgcGk7VicU0zE2OaYHbPnMVXbCn_?authuser=1#scrollTo=Ldo8UjEG5N5u.
- **Ética e Dados:** Em conformidade com as boas práticas de pesquisa, os dados coletados durante a demonstração passaram por um processo de anonimização prévia, removendo identificadores pessoais dos participantes antes do processamento.

4. Resultados

Neste capítulo, serão apresentados os dados e as estatísticas obtidas após a execução do pipeline de avaliação. Os resultados foram organizados em três seções: análise estatística das métricas nos dados reais e sintéticos, e as observações após a análise qualitativa manual.

4.1. Considerações Estatísticas

Para apresentar os dados, foram computadas estatísticas descritivas fundamentais para caracterizar o comportamento dos LLMs. Os indicadores principais foram:

- **Média:** Indica o desempenho central do modelo na métrica avaliada.
- **Mediana:** Utilizada para mitigar o impacto de valores extremos (*outliers*), comum em LLMs que podem oscilar entre respostas perfeitas e alucinações totais.
- **Desvio Padrão:** Medida de dispersão que indica a estabilidade operacional. Valores altos indicam comportamento menos previsível.
- **Mínimo e Máximo:** Definem a amplitude de qualidade observada nas amostras.

4.2. Desempenho Quantitativo: Dados Reais (Piloto)

O primeiro cenário de avaliação compreende as interações com usuários reais (N=26).

4.2.1. Faithfulness (Fidelidade)

A Tabela 1 apresenta os valores obtidos para a métrica de *Faithfulness* (Fidelidade).

Tabela 1. Estatísticas descritivas da métrica *Faithfulness* (dados reais).

Modelo	Registros	Média	Mediana	Desvio	Mínimo	Máximo
anthropic/claude-3.5-sonnet	1	0.2500	0.2500	—	0.2500	0.2500
openai/gpt-5-chat	5	0.2260	0.1250	0.2280	0.0000	0.5714
google/gemini-2.5-pro	20	0.1781	0.1389	0.2001	0.0000	0.6667

Observa-se que o **Gemini 2.5 Pro** foi o modelo predominante no roteamento, apresentando consistência moderada. O **GPT-5 Chat**, apesar da média superior, demonstrou maior variabilidade (desvio de 0.2280).

4.2.2. Answer Relevancy (Relevância)

Para a métrica de Relevância da Resposta (*Answer Relevancy*), os dados são detalhados na Tabela 2.

Tabela 2. Estatísticas descritivas da métrica *Answer Relevancy* (dados reais).

Modelo	Registros	Média	Mediana	Desvio	Mínimo	Máximo
google/gemini-2.5-pro	38	0.3044	0.2978	0.1817	0.0000	0.7690
anthropic/claude-3.5-sonnet	19	0.3271	0.3168	0.1934	0.0000	0.8167
openai/gpt-5-chat	23	0.3218	0.2963	0.1848	0.0714	0.6562

4.3. Desempenho Quantitativo: Dados Sintéticos (Estendido)

O segundo cenário apresenta os resultados do teste de estresse com dados sintéticos (N=252), ampliando a base estatística.

4.3.1. Faithfulness (Fidelidade)

A Tabela 3 apresenta os resultados de fidelidade neste cenário.

Tabela 3. Métrica *Faithfulness* com dados artificiais.

Modelo	Registros	Média	Mediana	Desvio	Mínimo	Máximo
google/gemini-2.5-pro	49	0.1726	0.1667	0.2017	0.0000	0.6667
anthropic/claude-3.5-sonnet	57	0.1874	0.0833	0.2321	0.0000	0.8421
openai/gpt-5-chat	52	0.1395	0.0000	0.1917	0.0000	0.6000

Os dados revelam três perfis distintos de distribuição de scores:

1. **Claude 3.5 Sonnet:** Apresentou o maior potencial (máximo de 0.84), mas com maior instabilidade.
2. **Gemini 2.5 Pro:** Demonstrou ser o mais robusto, com a maior mediana (0.1667).
3. **GPT-5 Chat:** Apresentou mediana zero nas interações sintéticas.

4.3.2. Answer Relevancy (Relevância)

Em contrapartida, na métrica de relevância (Tabela 4), observa-se uma alteração na liderança dos indicadores.

Tabela 4. Métrica *Answer Relevance* com dados artificiais.

Modelo	Registros	Média	Mediana	Desvio	Mínimo	Máximo
google/gemini-2.5-pro	79	0.4147	0.3812	0.1602	0.0000	0.8457
anthropic/claude-3.5-sonnet	93	0.3871	0.4407	0.2601	0.0000	0.8825
openai/gpt-5-chat	80	0.4637	0.4358	0.1718	0.1391	0.8694

4.4. Achados da Análise Qualitativa

A inspeção qualitativa manual, realizada por amostragem das interações, identificou padrões textuais específicos que permitem correlacionar o desempenho com as características de design dos modelos:

- **Padrão de Extrapolação (Claude):** A promessa de ”nuance e tom natural”deste modelo resultou, na prática, em uma tendência a ”ajudar demais”. Observou-se que o Claude frequentemente detalha pontos que não estavam no contexto original para tornar a conversa mais fluida, o que enriquece a experiência do usuário mas penaliza drasticamente o score de *Faithfulness* (alucinação extrínseca).
- **Padrão Pragmático (GPT-5):** Refletindo sua característica de ”sistema unificado”focado em resolução, as respostas mostraram-se diretas e utilitárias. O modelo priorizou responder à dúvida do usuário (Alta Relevância) mesmo quando o contexto recuperado era insuficiente, utilizando seu conhecimento interno em detrimento da restrição documental.
- **Padrão Formal (Gemini):** Alinhado à sua arquitetura de ”thinking model”com foco em contextos longos, o Gemini manteve um comportamento conservador. Suas respostas frequentemente copiavam a estrutura dos documentos recuperados, resultando na maior estabilidade de fidelidade observada.

5. Discussão

Nesta seção, interpretam-se os resultados apresentados anteriormente, comparando os cenários de teste e analisando o comportamento prático de cada modelo.

5.1. Convergência entre Cenários Real e Sintético

A análise conjunta dos dois testes (com usuários reais e com o robô simulado) mostra que o uso de dados sintéticos foi eficaz. Eles serviram para confirmar tendências que apareceram no teste piloto, mas que precisavam de mais dados para serem validadas:

- **Assimetria do Claude:** A instabilidade observada na única amostra real do Claude 3.5 se confirmou e se intensificou com o aumento do volume de dados, validando a hipótese de que seus picos de qualidade vêm acompanhados de variações bruscas (alto desvio padrão).
- **Previsibilidade do Gemini:** O comportamento ”médio e estável”do Gemini 2.5 Pro se manteve consistente em ambas as bases, validando-o como a opção conservadora.
- **Evolução do GPT-5:** O modelo mostrou que, embora tenha baixa fidelidade documental (ignora o contexto com frequência), sua capacidade de entregar respostas relevantes melhora substancialmente quando analisada em volume, sugerindo uma ”inteligência geral”superior à sua capacidade de recuperação estrita.

Essa convergência aumenta a confiança na interpretação global dos resultados e valida o uso da automação via `gpt-4o-mini` como um proxy eficaz para avaliação de sistemas conversacionais.

5.2. Correlação entre Capacidades Declaradas e Desempenho

Ao confrontar os resultados obtidos com as justificativas de seleção descritas na metodologia, é possível traçar uma correlação direta entre as capacidades arquiteturais de cada modelo e seu comportamento no pipeline RAG:

1. **OpenAI GPT-5-Chat:**A documentação promete redução de alucinações, mas no contexto do RAG, o modelo tendeu a confiar mais no seu próprio conhecimento do que nos documentos fornecidos. Por outro lado, a promessa de ”inteligência unificada”se confirmou na métrica de Relevância: ele foi o melhor em entender a intenção do usuário e entregar uma solução prática.
2. **Google Gemini 2.5 Pro:** A escolha deste modelo por sua capacidade de raciocínio (”thinking model”) provou-se correta. O processamento extra parece ajudar o modelo a se ater mais aos documentos enviados, resultando na maior estabilidade de fidelidade entre os testados.
3. **Anthropic Claude 3.5 Sonnet:** A característica de ”tom natural e humano”funcionou de forma contraditória. Para tornar a conversa mais fluida, o modelo frequentemente adicionou detalhes que não existiam na iniciativa original. Ou seja, a tentativa de ser mais ”humano”acabou gerando alucinações.

5.3. Síntese dos Perfis

Os resultados apontam para uma escolha clara entre **criatividade** e **precisão**. A Tabela 5 resume os pontos fortes e fracos identificados.

Tabela 5. Síntese do perfil dos modelos avaliados.

Modelo	Ponto Forte	Ponto Fraco
Claude 3.5 Sonnet	Qualidade de escrita (Máximos)	Instabilidade / Alucinação
GPT-5 Chat	Relevância e utilidade	Baixa fidelidade ao contexto
Gemini 2.5 Pro	Estabilidade e segurança	Menor brilho criativo

Esses perfis indicam aplicações distintas. Para sistemas críticos que não podem conter erros factuais (como na área jurídica), o `Gemini` é a escolha mais segura. Para assistentes focados em gerar ideias, o `GPT-5` oferece maior utilidade. Já o `Claude` exige camadas extras de verificação devido à sua instabilidade.

5.4. Limitações e Ameaças à Validade

É importante destacar as limitações deste trabalho. A quantidade de usuários reais foi reduzida (N=26), o que aumentou a dependência dos dados simulados para a análise.

Além disso, a ferramenta utilizada para avaliar as respostas (o ”Juiz”) é baseada no GPT-4o. Existe a possibilidade de um viés de avaliação, onde o juiz favorece as respostas do GPT-5 por terem um estilo de escrita semelhante, o que poderia inflar levemente suas notas de relevância.

Por fim, todos os dados coletados foram anonimizados, garantindo a privacidade dos participantes e a ética da pesquisa.

6. Conclusão e Trabalhos Futuros

Este trabalho teve como objetivo principal analisar como a escolha do Modelo de Linguagem (LLM) influencia a qualidade de um assistente virtual que usa a técnica RAG. Para isso, o sistema foi implementado na plataforma Inithub e testado tanto com estudantes reais quanto com dados simulados por computador.

Os resultados mostraram que não existe um modelo que seja o melhor em tudo. Percebeu-se que existe uma troca: ou o modelo é muito fiel ao documento, ou ele é mais criativo e útil. O `Google Gemini 2.5 Pro` se mostrou a opção mais estável e segura, ideal para quando o sistema não pode errar a informação do contexto. Já o `OpenAI GPT-5-Chat` foi o que melhor entendeu o que o usuário queria e deu as respostas mais diretas, mesmo que às vezes ele tenha ignorado um pouco os documentos para usar seu próprio conhecimento. O `Claude 3.5 Sonnet` escreve muito bem, mas seus resultados variaram muito, ora sendo excelentes, ora inventando informações, o que exige cuidado no seu uso.

Além disso, ficou comprovado que usar dados sintéticos (gerados por automação) é uma estratégia válida. Os testes feitos pelo robô mostraram resultados muito parecidos com os testes feitos por pessoas reais, o que ajuda muito a testar o sistema mais rápido antes de lançar para o público.

Como sugestão para trabalhos futuros, a principal ideia é deixar a plataforma disponível por mais tempo. Com mais pessoas usando e cadastrando iniciativas de verdade, teremos um banco de dados maior para confirmar se esses resultados se mantêm a longo prazo.

Outro passo importante seria criar um conjunto de ”respostas ideais”(gabarito) escritas por humanos. Hoje, nós avaliamos o sistema sem esse gabarito, mas se tivermos as respostas certas para comparar, poderemos usar outras métricas do Ragas que dão uma visão mais detalhada sobre a qualidade.

Por fim, recomenda-se testar novos modelos que estão surgindo no mercado, inclusive modelos de código aberto que podem ser mais baratos, e também testar novas formas de escrever os prompts (instruções) para tentar corrigir a instabilidade que encontramos no modelo Claude.

Referências

[Anthropic 2024] Anthropic (2024). Claude 3.5 sonnet. `https://www.anthropic.com/news/claude-3-5-sonnet`.

[Bengio et al. 1994] Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166.

[Brown et al. 2020] Brown, T. et al. (2020). Language models are few-shot learners. *Advances in Neural Information Processing Systems*.

[Gao et al. 2023] Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., and Wang, H. (2023). Retrieval-augmented generation for large language models: A survey. *arXiv preprint arXiv:2312.10997*.

[Gemini Team, Google 2025] Gemini Team, Google (2025). Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. Technical Report arXiv:2507.06261v5, Google.

[Hochreiter and Schmidhuber 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8):1735–1780.

[Ji et al. 2023] Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. (2023). Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38.

[Lewis et al. 2020] Lewis, P., Perez, E., Piktus, A., et al. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*.

[Liu et al. 2016] Liu, C.-W., Lowe, R., Serban, I., Noseworthy, M., Charlin, L., and Pineau, J. (2016). How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2122–2132.

[McKinsey & Company 2024] McKinsey & Company (2024). The state of ai in early 2024: Gen ai adoption spikes and starts to generate value. Technical report, McKinsey & Company. Acesso em: [Data de hoje].

[OpenAI 2025] OpenAI (2025). Introducing gpt-5: Our smartest, fastest, and most helpful model yet. `https://openai.com/pt-BR/index/introducing-gpt-5/`. Acessado em: 28 nov. 2025.

[Reimers and Gurevych 2019] Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP-IJCNLP 2019*, pages 3982–3992, Hong Kong, China.

[Shahul Es 2023] Shahul Es, Jithin James, L. E.-A. S. S. (2023). Ragas: Automated evaluation of retrieval augmented generation. arXiv preprint arXiv:2309.15217.

[Simon et al. 2024] Simon, S., Mailach, A., Dorn, J., and Siegmund, N. (2024). A methodology for evaluating rag systems: A case study on configuration dependency validation. *arXiv preprint arXiv:2410.08801*.

[Vaswani et al. 2017] Vaswani, A. et al. (2017). Attention is all you need. *Advances in Neural Information Processing Systems*.

[Zhao et al. 2023] Zhao, W. X. et al. (2023). A survey of large language models. *arXiv preprint arXiv:2303.18223*.