

**CENTRO DE COMPETÊNCIAS EMBRAPPI EM
TECNOLOGIAS IMERSIVAS (AKCIT)**
**PROGRAMA DE CAPACITAÇÃO E FORMAÇÃO EM TECNOLOGIAS
IMERSIVAS E IA**

**LARISSA NEPOMUCENO COSTA MARINHO
VANESSA SCHEMES MARTINS PINTO**

**IMPACTO DA PREPARAÇÃO E DA
RECUPERAÇÃO DO CORPUS SOBRE A
EFICIÊNCIA DE SISTEMAS RAG:**
**UM BENCHMARK PAREADO EM PERGUNTAS
MULTI-HOP**

Goiânia – GO

2026

**LARISSA NEPOMUCENO COSTA MARINHO
VANESSA SCHEMES MARTINS PINTO**

**IMPACTO DA PREPARAÇÃO E DA
RECUPERAÇÃO DO CORPUS SOBRE A
EFICIÊNCIA DE SISTEMAS RAG:
UM BENCHMARK PAREADO EM PERGUNTAS
MULTI-HOP**

Trabalho de Conclusão de Curso submetido ao Programa de Capacitação e Formação do Centro de Competências Embrapii em Tecnologias Imersivas (AKCIT) como requisito obrigatório para a obtenção do certificado de conclusão da especialização em Inteligência Artificial e Processamento de Linguagem Natural.

**Goiânia – GO
2026**

LARISSA NEPOMUCENO COSTA MARINHO
VANESSA SCHEMES MARTINS PINTO

**IMPACTO DA PREPARAÇÃO E DA
RECUPERAÇÃO DO CORPUS SOBRE A
EFICIÊNCIA DE SISTEMAS RAG:
UM BENCHMARK PAREADO EM PERGUNTAS
MULTI-HOP**

Trabalho de Conclusão de Curso submetido ao Programa de Capacitação e Formação do Centro de Competências Embrapii em Tecnologias Imersivas (AKCIT) como requisito obrigatório para a obtenção do certificado de conclusão da especialização em Inteligência Artificial e Processamento de Linguagem Natural.

Aprovado em: 23 de setembro de 2026.

BANCA EXAMINADORA

Prof(a). Dr(a). [Nome do(a) Orientador(a)] – Orientador(a) / Presidente
Centro de Competências Embrapii em Tecnologias Imersivas (AKCIT)

Prof(a). Dr(a). [Nome do 1º Avaliador(a)] – Membro Interno
Centro de Competências Embrapii em Tecnologias Imersivas (AKCIT)

Prof(a). Dr(a). [Nome do 2º Avaliador(a)] – Membro Externo
Universidade / Instituto Convidado

RESUMO

Introdução: Sistemas de Geração Aumentada por Recuperação (RAG) enviam documentos recuperados dentro do prompt a cada requisição, o que eleva o custo por consulta e a latência de resposta. A forma como esses documentos são preparados e recuperados antes da indexação é uma variável de projeto ainda pouco quantificada na literatura. **Objetivo:** Comparar, em benchmark experimental controlado, o efeito do fatiamento linear (Naive RAG), da segmentação por hierarquia Markdown (Structured RAG) e da recuperação hierárquica *parent-child* (Hierarchical RAG) sobre a qualidade da recuperação, a qualidade da resposta, a latência de processamento e o consumo de tokens. **Descrição detalhada:** Foram submetidas 180 questões do benchmark HotpotQA aos três cenários, em três repetições técnicas pareadas, totalizando 1.620 execuções em estação local dedicada equipada com GPU de 8 GB. Mantiveram-se constantes o modelo gerador Llama 3.1 8B (quantizado em 4 bits), o modelo de vetorização (*nomic-embed-text*), o banco vetorial (ChromaDB) e o conjunto de perguntas. A latência foi decomposta em busca vetorial, processamento do prompt (*prefill*) e decodificação autorregressiva. A qualidade das respostas foi mensurada por métricas lexicais e por protocolo cego de julgamento com três avaliadores independentes (um juiz humano de referência e dois modelos de fronteira de fornecedores distintos), cuja concordância inter-avaliador atingiu ICC(3,3) de 0,9833. Os contrastes empíricos foram estimados por inferência bayesiana pareada (Student-*t* e regressão logística), associados a Regiões de Equivalência Prática (ROPE) declaradas *a priori*. **Considerações finais:** As estratégias estruturadas reduziram a latência total de 8,40 s para cerca de 6,30 s (redução de 25%) e o volume de tokens de entrada em 33,3%, ambas com probabilidade posterior de 100%. Para o Hierarchical RAG, não foi detectada diferença de qualidade semântica em relação ao controle, mas a incerteza das estimativas não permite afirmar equivalência prática estrita. O Structured RAG apresentou resultados inferiores nas métricas descritivas, com evidência de redução em F1. Os resultados comprovam que expressivos ganhos de eficiência de engenharia podem ser viabilizados sem detecção de perda de qualidade no ambiente investigado.

Palavras-chave: Inteligência Artificial. Processamento de Linguagem Natural. Geração Aumentada por Recuperação. Engenharia de Dados Documentais. Benchmarking.

ABSTRACT

Introduction: Retrieval-Augmented Generation (RAG) systems inject retrieved external documents into the prompt at each request, increasing inference latency and operational cost. How documents are structured and retrieved prior to generation remains a design variable scarcely quantified in the empirical literature. **Objective:** To compare, under a controlled experimental benchmark, the effect of linear fixed-size chunking (Naive RAG), Markdown structural hierarchy segmentation (Structured RAG), and parent-child hierarchical retrieval (Hierarchical RAG) on retrieval efficacy, generation quality, hardware latency, and token consumption. **Detailed description:** A total of 180 HotpotQA questions were evaluated across all three architectures over three paired technical repetitions, totaling 1,620 inference runs on a dedicated local workstation equipped with an 8 GB GPU. The local generator model (Llama 3.1 8B, 4-bit quantized), the embedding model (*nomic-embed-text*), the vector database (ChromaDB), and the question set were strictly held constant. End-to-end latency was decomposed into vector search, prompt processing (*prefill*), and autoregressive token decoding. Answer quality was measured using lexical metrics and a triple-blind evaluation protocol with three independent judges (a human reference auditor and two frontier LLMs from distinct providers), achieving an inter-rater reliability ICC(3,3) of 0.9833. Contrasts were estimated via paired Bayesian inference (Student-*t* and logistic regressions) with Regions of Practical Equivalence (ROPE) specified *a priori*. **Final considerations:** Structured architectures reduced total latency from 8.40 s to approximately 6.30 s (a 25% speedup) and input prompt tokens by 33.3%, both with a posterior probability of 100%. For Hierarchical RAG, no statistically significant quality difference was detected relative to the control, though posterior uncertainty precludes asserting strict practical equivalence. Structured RAG exhibited inferior descriptive metrics, with marked reductions in lexical F1. Findings demonstrate that substantial hardware and volumetric efficiency gains can be achieved without observable degradation in generative semantic fidelity.

Keywords: Artificial Intelligence. Natural Language Processing. Retrieval-Augmented Generation. Document Data Engineering. Benchmarking.

LISTA DE FIGURAS

3.1	Estrutura de diretórios e organização modular do projeto.	21
3.2	Fluxo de dados da esteira experimental e rastreabilidade dos artefatos intermediários.	23
3.3	Diagrama arquitetural dos três cenários RAG avaliados.	24
4.1	Resumo comparativo do desempenho global entre os cenários RAG.	43
4.2	Distribuição empírica da latência total por arquitetura RAG.	44
4.3	Intervalos de mais alta densidade posterior (HDI 95%) e ROPE para as diferenças pareadas de latência.	46
4.4	Decomposição da latência média por fase do pipeline RAG.	47
4.5	Contrastes a posteriori das etapas do pipeline RAG (Prefill, Busca Vetorial e Geração) com intervalos HDI 95% e ROPE.	48
4.6	Forest plot dos intervalos de densidade mais alta (HDI 95%) e ROPE para tokens de entrada.	51
4.7	Consumo médio de tokens de prompt (Painel A) e correlação linear entre volume de tokens e tempo de prefill na GPU (Painel B).	53
4.8	Eficácia de recuperação de evidências (Painel A) e acurácia semântica por nível de dificuldade (Painel B).	58
4.9	Triangulação de notas do painel tripartite de juízes: matriz de correlação (Painel A), densidade bimodal (Painel B) e pontuação média contínua por cenário RAG (Painel C).	63
4.10	Telemetria temporal da GPU durante a execução do benchmark.	64

LISTA DE TABELAS

3.1	Ecossistema de <i>software</i> , bibliotecas e componentes da arquitetura experimental.	31
4.1	Estatísticas descritivas da latência total de inferência por arquitetura RAG ($N = 180$).	44
4.2	Estimativas a posteriori das diferenças pareadas de latência (Δ Tempo em segundos).	45
4.3	Decomposição média da latência por fase do pipeline RAG ($N = 180$). . . .	46
4.4	Métricas descritivas do consumo de tokens de entrada e eficiência de empacotamento ($N = 180$).	50
4.5	Estimativas a posteriori das diferenças pareadas de tokens de entrada (Δ Prompt Tokens).	51
4.6	Métricas globais de eficácia de recuperação e qualidade de geração ($N = 180$). .	54
4.7	Estimativas a posteriori das taxas de recuperação e acurácia gerativa. . . .	55
4.8	Eficácia de recuperação e acurácia gerativa estratificadas por nível de dificuldade ($n = 60$ por nível).	57
4.9	Confiabilidade inter-avaliadores e matriz de concordância entre os juízes ($N = 540$ respostas avaliadas)	61
4.10	Matriz de correlações pareadas de Pearson (r) e Spearman (r_s) entre os juízes triangulados.	61
4.11	Estatísticas descritivas das notas atribuídas pelos juízes e consolidação por cenário (0 a 10 pontos).	62
5.1	Síntese da validação experimental das hipóteses de pesquisa.	67

LISTA DE ABREVIATURAS E SIGLAS

AKCIT	Centro de Competências Embrapii em Tecnologias Imersivas
API	<i>Application Programming Interface</i>
BPE	<i>Byte-Pair Encoding</i>
CoT	<i>Chain-of-Thought</i> (Cadeia de Pensamento)
CPU	<i>Central Processing Unit</i> (Unidade Central de Processamento)
CUDA	<i>Compute Unified Device Architecture</i>
EM	<i>Exact Match</i> (Correspondência Exata)
EMBRAPII	Empresa Brasileira de Pesquisa e Inovação Industrial
ESS	<i>Effective Sample Size</i> (Tamanho Amostral Efetivo)
GPU	<i>Graphics Processing Unit</i> (Unidade de Processamento Gráfico)
HDI	<i>Highest Density Interval</i> (Intervalo de Mais Alta Densidade)
HNSW	<i>Hierarchical Navigable Small World</i>
HTML	<i>HyperText Markup Language</i>
ICC	<i>Intraclass Correlation Coefficient</i> (Coeficiente de Correlação Intraclasse)
IQR	<i>Interquartile Range</i> (Intervalo Interquartil)
KV-Cache	<i>Key-Value Cache</i>
LLM	<i>Large Language Model</i> (Modelo de Linguagem de Grande Escala)
MCMC	<i>Markov Chain Monte Carlo</i>
NLG	<i>Natural Language Generation</i>
NLP / PLN	<i>Natural Language Processing</i> / Processamento de Linguagem Natural
NUTS	<i>No-U-Turn Sampler</i>
OCR	<i>Optical Character Recognition</i>
OPEX	<i>Operational Expenditure</i> (Custo Operacional)
OR	<i>Odds Ratio</i> (Razão de Chances)
RAG	<i>Retrieval-Augmented Generation</i> (Geração Aumentada por Recuperação)
ROPE	<i>Region of Practical Equivalence</i> (Região de Equivalência Prática)
VRAM	<i>Video Random Access Memory</i> (Memória de Vídeo Dedicada)

SUMÁRIO

1	Introdução	7
1.1	Objetivos da Pesquisa	8
1.1.1	Objetivo Geral	8
1.1.2	Objetivos Específicos	8
1.2	Questão de Pesquisa e Hipóteses de Trabalho	9
1.3	Contribuições e Organização da Monografia	10
2	Fundamentação Teórica	12
2.1	Modelos Neurais Autorregressivos e Dinâmica de Atenção	12
2.2	O Paradigma RAG e a Modelagem de Memória Não Paramétrica	13
2.3	Engenharia de Layout Documental e Estruturação de Texto	15
2.4	Indexação Vetorial e Busca Aproximada em Grafos (ANN)	16
2.5	Raciocínio Multi-Hop e Degradação Contextual (Lost in the Middle)	17
2.6	Avaliação Semântica Triangulada e Inferência Bayesiana	18
3	Realização do Experimento	20
3.1	Arquitetura do Experimento e Estratégias RAG Comparadas	20
3.1.1	Visão Geral e Arquitetura Modular do Software	20
3.1.2	Fluxo de Dados Ponta a Ponta	22
3.1.3	Caracterização dos Cenários RAG e Desenho de Ablação	22
3.1.4	Resolução Hierárquica (<i>parent_fetch</i>)	25
3.1.5	Gestão de Contexto e Orçamento Dinâmico de Tokens	25
3.2	Amostragem Estratificada e Seleção das Questões	26
3.2.1	Fonte dos Dados e Recorte Temático	26
3.2.2	Estratificação por Dificuldade e Sorteio Determinístico	27
3.2.3	Validação Humana e Protocolo de Reposição 1:1	27
3.3	CONSTRUÇÃO, SANITIZAÇÃO E AUDITORIA DO CORPUS	28

3.3.1	Resgate Histórico e Mitigação de <i>Data Drift</i>	28
3.3.2	Sanitização Prévia do DOM e Paridade de Formatos	29
3.3.3	Auditoria Automatizada e Limite Empírico de Cobertura	29
3.4	Ambiente Operacional, Infraestrutura de Hardware e Software	30
3.4.1	Especificações de Hardware e Isolamento Experimental	30
3.4.2	Modelos de Linguagem, Embeddings e Parametrização Técnica	32
3.5	PROTOCOLO DE EXECUÇÃO PAREADA, ALEATORIZAÇÃO E CON- TROLE DE VIÉS	33
3.5.1	Desenho Experimental Pareado e Volume Amostral	33
3.5.2	Aleatorização de Ordem e Rodadas Globais Intercaladas	33
3.5.3	Regime Estrito de Concorrência Serial ($N = 1$)	34
3.5.4	Eliminação do Efeito de Carga Inicial (<i>Warm-up Protocol</i>)	34
3.5.5	Telemetria Contínua de Infraestrutura em Segundo Plano	34
3.6	INDICADORES DE DESEMPENHO, MÉTRICAS DE TELEMETRIA E MAPEAMENTO DE DADOS	35
3.6.1	Métricas de Eficácia da Recuperação de Contexto	35
3.6.2	Métricas Lexicais Secundárias de Qualidade da Resposta (Análise Descritiva)	35
3.6.3	Decomposição Temporal e Telemetria de Latência	36
3.6.4	Alocação de Contexto e Métricas de Custo Computacional	37
3.6.5	Consolidação do Dataset Mestre e Estrutura de Artefatos	37
3.7	PROTOCOLO DE AVALIAÇÃO CEGA E VALIDAÇÃO DA QUALIDADE SEMÂNTICA	38
3.7.1	Superação das Limitações Lexicais via Julgamento Semântico e Definição de H_2	38
3.7.2	Protocolo de Cegamento e Anonimização	38
3.7.3	Painel de Triangulação de Três Juízes	39
3.7.4	Concordância Interavaliadores e Salvaguarda Metodológica	40
3.7.5	Modelagem Inferencial Bayesiana e Validação das Hipóteses	40
3.8	REPRODUTIBILIDADE, AUDITORIA E ASPECTOS ÉTICOS	41

3.8.1	Declaração de Uso de Ferramentas de IA na Implementação Computacional	42
4	Resultados e Discussão	43
4.1	ANÁLISE COMPARATIVA DA LATÊNCIA DE RESPOSTA E EFICIÊNCIA DE HARDWARE	43
4.1.1	Desempenho Global da Latência: Comparativo entre Arquiteturas .	43
4.1.2	Modelagem Bayesiana da Latência Pareada	44
4.1.3	Decomposição da Latência por Etapas do Pipeline: A Causa Raiz do Ganho	46
4.1.4	Síntese da Discussão de Latência	49
4.2	ANÁLISE DO CONSUMO DE TOKENS, CARGA DE CONTEXTO E EFICIÊNCIA DE EMPACOTAMENTO	49
4.2.1	Carga de Tokens Injetada no Prompt por Arquitetura	49
4.2.2	Modelagem Bayesiana do Consumo de Tokens Pareado	50
4.2.3	Fragmentação de Documentos e Dinâmica de Deduplicação de Pais	52
4.2.4	Relação Causal entre Consumo de Tokens e Latência de Prefill . . .	52
4.2.5	Mitigação do Efeito <i>Lost-in-the-Middle</i> e Implicações Econômicas .	53
4.3	EFICÁCIA DE RECUPERAÇÃO, ACURÁCIA SEMÂNTICA E ESTRATIFICAÇÃO POR COMPLEXIDADE	53
4.3.1	Desempenho Global de Recuperação e Geração	54
4.3.2	Modelagem Bayesiana Logística da Taxa de Acerto e Hit Rate . . .	55
4.3.3	O Paradoxo da Recuperação: Fragmentos Ruidosos vs. Integridade Contextual	55
4.3.4	Estratificação do Desempenho por Nível de Complexidade (<i>Easy, Medium, Hard</i>)	57
4.3.5	Limitações do Retriever Puro em Raciocínio Multi-Hop e Efeito Gargalo	59
4.4	CONFIABILIDADE METODOLÓGICA E TRIANGULAÇÃO DOS JUÍZES	59
4.4.1	Protocolo Cego de Avaliação e Padronização de Rubrica	60
4.4.2	Confiabilidade Inter-Avaliador e Coeficiente de Correlação Intraclass (ICC)	61

4.4.3	Distribuição das Notas Contínuas e Análise por Faixa de Pontuação	61
4.4.4	Análise Qualitativa de Casos Limítrofes e Resolução de Discrepâncias	62
4.4.5	Validação Metodológica do Paradigma LLM-as-a-Judge	63
4.5	TELEMETRIA DE HARDWARE E AUDITABILIDADE	64
5	Conclusão	65
5.1	SÍNTESE DOS RESULTADOS E RESPOSTA À PERGUNTA DE PESQUISA	65
5.2	VALIDAÇÃO FORMAL DAS HIPÓTESES DE PESQUISA	66
5.2.1	Discussão Epistemológica dos Vereditos	66
5.3	CONTRIBUIÇÕES PARA A ENGENHARIA DE SOFTWARE E SISTEMAS DE IA	68
5.4	LIMITAÇÕES DO ESTUDO	68
5.5	TRABALHOS FUTUROS E DIRETRIZES DE CONTINUIDADE	69

INTRODUÇÃO

A convergência recente do Processamento de Linguagem Natural em torno de Modelos de Linguagem de Grande Porte (*Large Language Models*, LLMs) consolidou o uso de arquiteturas neurais autorregressivas baseadas em autoatenção (VASWANI et al., 2017; BROWN et al., 2020). Apesar da capacidade desses modelos em tarefas complexas, sua aplicação em sistemas de software enfrenta limitações próprias do conhecimento paramétrico, em especial a alucinação factual (JI et al., 2023; HUANG et al., 2023) e a defasagem temporal decorrente da estaticidade dos pesos após o pré-treinamento.

Para contornar a dependência exclusiva da memória paramétrica, estabeleceu-se o paradigma de Geração Aumentada por Recuperação (*Retrieval-Augmented Generation*, RAG) (LEWIS et al., 2020). Essa abordagem desacopla o modelo gerador da base de conhecimento e injeta fragmentos de documentos externos diretamente no prompt a cada requisição, ancorando as respostas em evidências factuais sem a necessidade de re-treinamento ou ajuste fino (*fine-tuning*).

A operação de sistemas RAG em ambientes de produção, contudo, introduz desafios práticos de engenharia de software e infraestrutura computacional:

1. A sobrecarga durante o processamento do prompt (*prefill*) na Unidade de Processamento Gráfico (GPU), etapa em que a ingestão de extensos volumes contextuais, o cálculo das matrizes de autoatenção e a alocação do cache de chaves e valores (*KV cache*) concentram o gargalo temporal da inferência (DAO et al., 2022; KWON et al., 2023);
2. O custo operacional recorrente gerado pelo envio sistemático de fragmentos textuais redundantes, encarecendo consultas em serviços tarifados por volume de entrada ou saturando a memória de vídeo (VRAM) disponível em estações de trabalho locais (BALAGUER et al., 2024);
3. A degradação na geração de respostas induzida pela dispersão atencional (*Lost in the Middle*), decorrente de estratégias de fatiamento linear cego (*Naive RAG*) que rompem a sintaxe de parágrafos, listas e tabelas, diluindo as evidências úteis ao longo de passagens extensas e ruidosas (LIU et al., 2024).

Em ambientes produtivos, a viabilidade de sistemas RAG depende tanto da fidelidade das respostas quanto de limites rígidos de latência e custo computacional. Trabalhos

recentes têm priorizado arquiteturas de alta complexidade, como sistemas baseados em grafos de conhecimento (*Graph RAG*) (EDGE et al., 2024) ou agentes com busca recursiva (GAO et al., 2023). Essas abordagens, no entanto, introduzem sobrecargas expressivas de indexação assíncrona e processamento. Em contrapartida, o impacto de intervenções mais simples na etapa de preparação do corpus permanece pouco quantificado na literatura.

Avaliar o fatiamento pré-indexação por meio de conversores de layout, como o Docling (AUER et al., 2024), e de mecanismos de busca hierárquica em dois níveis (mantendo fixos o hardware, o modelo gerador e o modelo de vetorização), permite medir diretamente os compromissos entre tempo de execução em GPU, consumo de tokens e qualidade da resposta. Reduzir o volume de contexto enviado ao gerador diminui a ocupação de memória de vídeo e viabiliza a execução estável de pipelines RAG mesmo em máquinas locais com recursos limitados.

1.1 OBJETIVOS DA PESQUISA

1.1.1 Objetivo Geral

Avaliar comparativamente, por meio de um benchmark experimental quantitativo e controlado, o impacto de três estratégias de preparação e recuperação do corpus (*Naive RAG*, *Structured RAG* e *Hierarchical RAG*) sobre a eficácia de recuperação, a qualidade das respostas, a latência computacional e o consumo de tokens, mantendo estritamente constantes o modelo gerador local (Llama 3.1 8B quantizado em quatro bits), o modelo de vetorização (*nomic-embed-text*), o banco vetorial (ChromaDB) e o conjunto de consultas.

1.1.2 Objetivos Específicos

1. Extrair e auditar 180 questões enciclopédicas de raciocínio *multi-hop* a partir do benchmark *HotpotQA*, distribuídas uniformemente entre os estratos nativos de complexidade (60 fáceis, 60 médias e 60 difíceis);
2. Construir e congelar historicamente a base de conhecimento com 1.780 artigos integrais da Wikipédia indexados antes da data de corte do conjunto de dados (01/10/2017), gerando representações paralelas em texto plano (Grupo 1) e em Markdown hierárquico via Docling (Grupos 2 e 3);
3. Implementar o ambiente experimental em coleções estanques no ChromaDB, isolando o fatiamento linear em blocos de 512 tokens (Grupo 1), a segmentação delimitada por seções (Grupo 2) e o mecanismo hierárquico de busca em nós-filho de 128 tokens com resgate determinístico de seções-pai completas e deduplicação (Grupo 3);

4. Executar 1.620 ensaios de inferência pareados e serializados sob regime de concorrência unitária ($180 \text{ questões} \times 3 \text{ estratégias} \times 3 \text{ repetições técnicas}$), registrando a telemetria térmica da GPU e a latência decomposta em busca vetorial, processamento do prompt (*prefill*) e decodificação autorregressiva;
5. Submeter as 540 respostas únicas a protocolo de avaliação cega tripartite (composto por auditoria humana de referência e pelos modelos GPT-5 e Gemini 3.7 Pro), mensurando a consistência das notas por meio do Coeficiente de Correlação Intraclass $ICC(3, 3)$;
6. Estimar os contrastes pareados entre as arquiteturas por meio de modelagem bayesiana robusta e logística, determinando intervalos de mais alta densidade posterior (HDI 95%), probabilidades de superioridade e aderência a Regiões de Equivalência Prática (ROPE) declaradas *a priori*.

1.2 QUESTÃO DE PESQUISA E HIPÓTESES DE TRABALHO

A investigação orienta-se pela seguinte questão central:

Sob condições estritamente controladas de modelo gerador local (Llama 3.1 8B Q4_K_M), modelo de vetorização (*nomic-embed-text*), banco vetorial (ChromaDB) e conjunto de consultas (*HotpotQA*, $N = 180$), qual é o impacto de três estratégias de preparação e recuperação do corpus (*Naive RAG*, *Structured RAG* e *Hierarchical RAG*) sobre a eficácia de recuperação, a qualidade das respostas, a latência computacional e o consumo de tokens?

Com base na premissa de que a preservação da estrutura sintática e a granularidade do particionamento influenciam a densidade informacional do contexto recuperado, foram formuladas oito hipóteses experimentais direcionadas, comparando as abordagens estruturadas ao grupo de controle linear (*Naive RAG*, G1):

- H_1 (Eficácia de recuperação de evidências): a segmentação por seções no *Structured RAG* (H_{1a}) e a recuperação em dois níveis no *Hierarchical RAG* (H_{1b}) aumentam a taxa de resgate dos fatos de suporte oficiais ($Recall@20$) em relação ao *Naive RAG*;
- H_2 (Qualidade gerativa da resposta): o *Structured RAG* (H_{2a}) e o *Hierarchical RAG* (H_{2b}) aumentam a acurácia semântica das respostas geradas pelo modelo frente ao *Naive RAG*, conforme avaliação do painel cego de juízes;
- H_3 (Latência computacional de execução): o *Structured RAG* (H_{3a}) e o *Hierarchical RAG* (H_{3b}) reduzem o tempo total de resposta de ponta a ponta em relação ao *Naive RAG*;

- H_4 (Carga de contexto e consumo de tokens): o *Structured RAG* (H_{4a}) e o *Hierarchical RAG* (H_{4b}) reduzem o volume de tokens de entrada injetados no modelo gerador em relação ao *Naive RAG*.

Os critérios de decisão apoiam-se na combinação do Intervalo de Mais Alta Densidade Posterior a 95% (HDI 95%) com Regiões de Equivalência Prática (ROPE) declaradas *a priori* para as variáveis contínuas, e na probabilidade *a posteriori* de superioridade para as métricas sem faixa prévia.

1.3 CONTRIBUIÇÕES E ORGANIZAÇÃO DA MONOGRAFIA

As principais contribuições deste estudo para a Engenharia de Software e o Processamento de Linguagem Natural compreendem:

- Decomposição e quantificação dos fatores de latência em sistemas RAG: demonstração empírica de que a economia temporal proporcionada pelas arquiteturas estruturadas decorre integralmente do alívio na fase de processamento do prompt (*prefill*) na GPU (3,75 s no *Hierarchical RAG* contra 5,89 s no *Naive RAG*), mantendo-se equivalentes os tempos de busca vetorial em grafo e de decodificação autorregressiva;
- Eficiência volumétrica de tokens sem degradação de resposta: evidencição de que a recuperação hierárquica em dois níveis (*parent-child*) reduz em 33,3% o consumo de tokens de entrada frente ao particionamento linear (5.082 contra 7.621 tokens), sem diferença estatisticamente demonstrável na qualidade gerativa final (16,67% contra 14,44% em acurácia semântica e F_1 -score equivalente de 0,1458 contra 0,1399);
- Preservação da integridade sintática e mitigação de perdas por saturação de contexto: comprovação de que o resgate de seções-pai completas com deduplicação elimina a fragmentação textual imposta por políticas gulosas de empacotamento (*greedy packing*), registrando zero corte parcial ou descarte de blocos, em contraste com a média de 2,15 blocos parciais e 1,42 blocos descartados por consulta observada no *Naive RAG*;
- Pipeline experimental aberto, padronizado e auditável: disponibilização pública do arcabouço metodológico integral, abrangendo os scripts de automação do benchmark, as coleções vetoriais, os registros relacionais em SQLite, as trilhas de telemetria térmica da GPU e as matrizes do protocolo cego de triangulação inter-avaliadores.

O trabalho organiza-se em cinco capítulos. O Capítulo 1 introduz o problema de pesquisa, a justificativa técnica, os objetivos delimitados e as hipóteses experimentais

avaliadas. O Capítulo 2 revisa a fundamentação teórica pertinente, abrangendo arquiteturas neurais autorregressivas, mecanismos de autoatenção, o paradigma RAG, engenharia de layout documental e protocolos de validação semântica baseados em painéis de juízes.

O Capítulo 3 detalha a metodologia experimental: descreve a infraestrutura de hardware dedicada, o protocolo de congelamento histórico do corpus, as particularidades de implementação das três estratégias avaliadas, as rotinas de telemetria e o desenho do julgamento cego tripartite. O Capítulo 4 expõe a análise dos resultados empíricos, examinando os contrastes de latência computacional, o consumo de tokens, a dinâmica de empacotamento, a eficácia de recuperação e a modelagem inferencial bayesiana da acurácia semântica. Por fim, o Capítulo 5 valida formalmente as hipóteses de pesquisa, discute as contribuições para a engenharia de sistemas de software, pondera as ameaças à validade e indica direções para trabalhos futuros.

FUNDAMENTAÇÃO TEÓRICA

Este capítulo estabelece o arcabouço conceitual e formal necessário para compreensão do experimento e interpretação de seus resultados. A discussão estrutura-se a partir dos princípios operacionais dos Modelos de Linguagem de Grande Porte e da física de execução da autoatenção, estendendo-se à formulação do paradigma de Geração Aumentada por Recuperação (RAG). Em seguida, abordam-se as estratégias de engenharia de layout documental, os mecanismos de indexação vetorial aproximada em grafos e as restrições cognitivas associadas ao raciocínio *multi-hop* sob contextos extensos. O capítulo encerra-se com a fundamentação dos protocolos de avaliação semântica por painel de juízes e dos métodos de inferência bayesiana adotados na análise estatística dos dados.

2.1 MODELOS NEURAI AUTOREGRESSIVOS E DINÂMICA DE ATENÇÃO

A arquitetura *Transformer* (VASWANI et al., 2017) baseia-se em mecanismos de atenção para mapear dependências estatísticas entre tokens sem recorrer a estruturas de recorrência temporal ou convoluções. Em modelos autorregressivos, a distribuição de probabilidade do próximo elemento da sequência é condicionada unicamente aos tokens antecedentes.

No mecanismo de autoatenção por produto escalar escalonado (*Scaled Dot-Product Attention*), a sequência de entrada de comprimento N é projetada linearmente nas matrizes de consultas $Q \in \mathbb{R}^{N \times d_k}$, chaves $K \in \mathbb{R}^{N \times d_k}$ e valores $V \in \mathbb{R}^{N \times d_v}$, onde d_k denota a dimensionalidade do subespaço vetorial. A matriz de afinidade atencional e a representação de saída são obtidas analiticamente por:

$$\text{Attention}(Q, K, V) = \text{softmax} \left(\frac{QK^T}{\sqrt{d_k}} \right) V, \quad (2.1)$$

em que o produto QK^T estabelece uma matriz de compatibilidade de dimensão $N \times N$, resultando em uma complexidade assintótica temporal e espacial quadrática $\mathcal{O}(N^2)$ em relação à extensão do contexto processado. A atenção em múltiplas cabeças (*Multi-Head Attention*) estende esse cálculo ao particionar as representações em h projeções lineares independentes, viabilizando o aprendizado conjunto de correlações em distintos subespaços latentes (VASWANI et al., 2017).

Em nível de infraestrutura computacional, o ciclo de inferência desses modelos divide-se em duas fases físicas com regimes operacionais distintos (DAO et al., 2022; KWON et al., 2023; POPE et al., 2023):

1. Processamento do prompt (*prefill*): todos os tokens do contexto de entrada são computados em paralelo na Unidade de Processamento Gráfico (GPU). Esta etapa é caracterizada por operações de multiplicação matricial geral (*General Matrix Multiply*, GEMM), operando em regime limitado por processamento (*compute-bound*) nos núcleos tensoriais, momento em que são alocados os tensores do cache de chaves e valores (*KV cache*);
2. Decodificação autorregressiva (*decoding*): os tokens de saída são gerados sequencialmente, de forma univariada. A cada passo gerativo, apenas as projeções de chave e valor do token recém-emitido são calculadas e incorporadas ao *KV cache*. A taxa de emissão é limitada pela largura de banda de memória (*memory bandwidth-bound*), pois cada token requer a transferência completa dos parâmetros do modelo e do histórico do *KV cache* da memória de vídeo (VRAM) para as unidades funcionais de cálculo (SHAZEER, 2019; PATEL et al., 2024).

Essa assimetria operacional esclarece a dinâmica de latência em sistemas generativos: variações no volume do contexto recuperado incidem diretamente sobre o tempo gasto na fase de *prefill*, mantendo o custo temporal de decodificação por token dependente apenas da extensão da resposta produzida.

2.2 O PARADIGMA RAG E A MODELAGEM DE MEMÓRIA NÃO PARAMÉTRICA

A retenção de informações em um Modelo de Linguagem de Grande Porte ocorre unicamente por meio do ajuste de pesos sinápticos durante a etapa de pré-treinamento, constituindo a denominada memória paramétrica (PETRONI et al., 2019; ROBERTS; RAFFEL; SHAZEER, 2020). Embora arquiteturas modernas retenham volumes expressivos de fatos, a dependência isolada dessa representação impõe limites práticos severos ao projeto de software:

1. A ocorrência de alucinações factuais (*hallucinations*), caracterizadas pela emissão de sequências sintaticamente corretas e coerentes, porém desprovidas de sustentação empírica ou incompatíveis com o conhecimento factual (JI et al., 2023; HUANG et al., 2023);

2. A defasagem temporal estática (*knowledge cutoff*), determinada pela incapacidade estrutural de acessar eventos, dados proprietários ou alterações documentais posteriores ao fechamento do corpus de treinamento (LAZARIDOU et al., 2021);
3. A opacidade epistemológica e a ausência de trilha de auditoria, uma vez que a distribuição paramétrica difusa inviabiliza rastrear de forma determinística quais passagens textuais subsidiaram a resposta produzida (RIBEIRO; SINGH; GUESTRIN, 2016; DOSHI-VELEZ; KIM, 2017).

Para mitigar tais limitações sem recorrer ao re-treinamento contínuo de pesos, Lewis et al. (2020) introduziram o paradigma de Geração Aumentada por Recuperação (*Retrieval-Augmented Generation*, RAG), combinando um componente de recuperação densa com um decodificador autorregressivo condicionado a uma memória não paramétrica externa.

Sob essa formalização, dado um enunciado de entrada x , o recuperador extrai os fragmentos textuais pertinentes $z \in \text{top-}k(P(z|x))$ a partir de uma base indexada $\mathcal{D}$. A probabilidade condicional da sequência de saída $y = (y_1, \dots, y_T)$ sob a formulação por token (*RAG-Token*) é obtida marginalizando a geração sobre os k documentos recuperados:

$$P(y|x) = \sum_{z \in \text{top-}k} P(z|x) \prod_{t=1}^T P(y_t|x, z, y_{1:t-1}). \quad (2.2)$$

Nessa expressão, $P(z|x)$ descreve a probabilidade de resgate topológico no espaço vetorial denso, enquanto $P(y_t|x, z, y_{1:t-1})$ corresponde à probabilidade gerativa condicionada simultaneamente ao enunciado, ao histórico autorregressivo e aos fragmentos externos z injetados no contexto.

A taxonomia moderna categoriza a evolução desses sistemas em três gerações arquiteturais (GAO et al., 2023; YU et al., 2024):

1. *Naive RAG*: fluxo estático e sequencial composto por fatiamento linear fixo do corpus, busca vetorial direta e concatenação bruta dos fragmentos recuperados no prompt gerador, desprovido de mecanismos de adaptação estrutural (BARNETT et al., 2024);
2. *Advanced RAG*: arquitetura que incorpora etapas intermediárias de otimização de pré-recuperação (como expansão e reescrita de consultas) e de pós-recuperação (como reclassificação densa por modelos *cross-encoder* e compressão de contexto) (MA et al., 2023; XU et al., 2023);
3. *Modular RAG*: arquitetura configurável em módulos independentes que permite

orquestração dinâmica, roteamento condicional entre diferentes fontes e separação formal entre escalas de indexação e escalas de geração via recuperação hierárquica.

A transição entre essas abordagens define compromissos diretos de engenharia entre assertividade factual, latência global de inferência e ocupação da memória de vídeo (VRAM) dos aceleradores gráficos dedicados.

2.3 ENGENHARIA DE LAYOUT DOCUMENTAL E ESTRUTURAÇÃO DE TEXTO

A etapa de particionamento textual (*chunking*) define a granularidade das unidades informacionais submetidas à indexação vetorial e ao gerador. O fatiamento linear tradicional, baseado em janelas fixas de caracteres ou tokens com sobreposição estática (*sliding window*), ignora a semântica do documento e introduz três distorções estruturais (AUER et al., 2024):

1. A ruptura sintática e correferencial, caracterizada pela interrupção arbitrária de períodos gramaticais, o que fragmenta orações coordenadas ou subordinadas e desfaz anáforas necessárias à compreensão do texto;
2. A desestruturação de dados tabulares e relacionais, provocada pelo particionamento cego de linhas e colunas, transformando tabelas estruturadas em sequências textuais desconexas para os modelos de vetorização e de linguagem;
3. A supressão da hierarquia discursiva, ocasionada pelo descarte de títulos de seções e subtítulos estruturais, desprovendo os parágrafos recuperados do escopo temático original definido no documento-fonte.

Para mitigar essas limitações, a Análise de Layout Documental (*Document Layout Analysis*, DLA) aplica modelos de visão computacional e algoritmos de análise morfológica para reconhecer a tipologia funcional dos elementos constituintes da página, como cabeçalhos, parágrafos, listas e tabelas (ZHONG; TANG; YEPES, 2019; WANG et al., 2022). Ferramentas de conversão estrutural, a exemplo do Docling (AUER et al., 2024), convertem formatos heterogêneos em representações hierárquicas em Markdown, preservando as fronteiras semânticas naturais dos blocos textuais e a geometria bidimensional de estruturas relacionais.

Essa integridade estrutural viabiliza o desacoplamento das escalas de recuperação e de geração mediante estratégias em dois níveis (*Parent-Child Retrieval*). O modelo contorna o compromisso clássico entre especificidade e contexto:

- Nós-filho (*child chunks*): fragmentos compactos (como blocos de 128 tokens) otimizados para maximizar a densidade semântica e a acurácia da busca aproximada por vizinhos mais próximos no espaço vetorial denso;
- Nós-pai (*parent sections*): seções documentais íntegras associadas aos nós-filho, resgatadas deterministicamente a partir de identificadores relacionais no momento da inferência, fornecendo ao modelo gerador o contexto narrativo completo da passagem.

Ao recuperar as seções-pai completas e aplicar algoritmos de deduplicação idempotente de metadados, o pipeline mitiga a redundância informacional e elimina truncamentos arbitrários impostos por limites de orçamento de contexto.

2.4 INDEXAÇÃO VETORIAL E BUSCA APROXIMADA EM GRAFOS (ANN)

A recuperação densa baseia-se no mapeamento de sequências textuais em vetores contínuos de dimensão fixa $\mathbf{e} \in \mathbb{R}^d$, projetando passagens semanticamente afins em regiões de proximidade no espaço métrico latente (REIMERS; GUREVYCH, 2019; MUENNIGHOFF et al., 2023).

Ao contrário de abordagens léxicas baseadas em frequência de termos, como o BM25 (ROBERTSON; ZARAGOZA, 2009), modelos de vetorização densa com arquitetura de codificador duplo (*bi-encoder*), a exemplo do *nomic-embed-text* (NUSSBAUM et al., 2024), são otimizados por perdas contrastivas para capturar correlações latentes independentemente de sobreposição vocabular estrita. A proximidade semântica entre dois vetores $\mathbf{u}$ e $\mathbf{v}$ é comumente mensurada pela Similaridade por Cosseno:

$$\text{Sim}_{\cos}(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|_2 \|\mathbf{v}\|_2}. \quad (2.3)$$

Quando os vetores são previamente normalizados na norma euclidiana ($\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$), a similaridade angular coincide com o produto escalar interno $\mathbf{u} \cdot \mathbf{v}$, e a distância euclidiana quadrática decorre diretamente da relação $\|\mathbf{u} - \mathbf{v}\|_2^2 = 2(1 - \mathbf{u} \cdot \mathbf{v})$. A otimização geométrica do resgate resume-se, portanto, ao cálculo de produtos internos em alta dimensão.

Em bases com milhares de passagens, o cálculo exaustivo de distâncias (*brute-force k-NN*) exibe complexidade linear $\mathcal{O}(N \cdot d)$, tornando-se computacionalmente custoso em tempo de inferência interativa. Para mitigar esse custo, sistemas de gerenciamento de dados vetoriais, como o ChromaDB, empregam algoritmos de busca aproximada por vizinhos mais próximos (*Approximate Nearest Neighbors*, ANN) implementados sobre motores de

grafos como o `hnswlib` (PAN et al., 2024; HAN et al., 2023).

Dentre essas estruturas, destaca-se o grafo de Pequenos Mundos Navegáveis Hierárquicos (*Hierarchical Navigable Small World*, HNSW) (MALKOV; YASHUNIN, 2020). O algoritmo constrói uma hierarquia estratificada de grafos esparsos de proximidade: as camadas superiores contêm poucos vértices com arestas de longo alcance para travessia rápida do espaço métrico, enquanto as camadas inferiores aumentam a densidade de conexões para refinamento local. A navegação descende pelas camadas com complexidade assintótica temporal de busca $\mathcal{O}(\log N)$. Esse comportamento assintótico esclarece por que a etapa de recuperação vetorial atinge latências na escala de milissegundos e não constitui o gargalo temporal da inferência em arquiteturas RAG locais.

2.5 RACIOCÍNIO MULTI-HOP E DEGRADAÇÃO CONTEXTUAL (LOST IN THE MIDDLE)

Tarefas de resposta a perguntas em passo único (*single-hop*) apoiam-se na identificação de um fato isolado contido em uma única passagem textual. Em contrapartida, tarefas de raciocínio em múltiplos passos (*Multi-Hop Question Answering*) exigem que o sistema recupere, encadeie e sintetize evidências complementares distribuídas em múltiplos documentos independentes (WELBL; STENETORP; RIEDEL, 2018; YANG et al., 2018). Do ponto de vista representacional, essas tarefas modelam grafos de evidências latentes, nos quais a falha na recuperação de um único nó inviabiliza a resolução da cadeia inferencial.

Essas consultas manifestam-se predominantemente em duas estruturas relacionais:

1. Consultas em ponte (*bridge questions*), nas quais a entidade que soluciona o problema depende da identificação e resolução intermediária de um elemento relacional contido em uma passagem preliminar;
2. Consultas comparativas (*comparison questions*), que demandam a extração simultânea de atributos análogos pertencentes a duas ou mais entidades distintas, confrontando-os sob critérios temporais, numéricos ou conceituais.

O benchmark *HotpotQA* (YANG et al., 2018) estabeleceu-se como referência metodológica para tarefas dessa natureza. Em sua configuração com distratores (*distractor setting*), cada consulta é acompanhada por dez artigos da Wikipédia: dois artigos de suporte contendo as sentenças comprobatórias requeridas pelo gabarito e oito artigos distratores correlatos, inseridos propositalmente para avaliar a robustez do recuperador contra correspondências léxicas e semânticas espúrias. Para assegurar a validade interna do experimento, torna-se necessário mitigar o desvio editorial (*editorial drift*) da Wikipédia

(YAO et al., 2021) mediante o congelamento do corpus na data de referência do conjunto de dados (01/10/2017), prevenindo discrepâncias entre as evidências históricas e versões alteradas posteriormente.

Ao agrupar múltiplos fragmentos no contexto de entrada para suprir a demanda informativa do raciocínio *multi-hop*, os sistemas defrontam-se com a limitação atencional conhecida como *Lost in the Middle* (LIU et al., 2024). Essa assimetria operacional decorre da distribuição não uniforme dos pesos de autoatenção ao longo de janelas extensas: modelos autorregressivos demonstram maior facilidade em integrar dados alocados nas posições iniciais (*primacy bias*) e finais (*recency bias*) do prompt, originando uma curva de desempenho em formato de “U”. Conforme passagens redundantes e ruídos de indexação preenchem a região intermediária do contexto, a dispersão da atenção eleva o risco de omissão das evidências críticas, comprometendo a fidelidade factual da decodificação gerativa (XIAO et al., 2023).

2.6 AVALIAÇÃO SEMÂNTICA TRIANGULADA E INFERÊNCIA BAYESIANA

A quantificação do desempenho em sistemas de Processamento de Linguagem Natural apoiou-se tradicionalmente em métricas determinísticas de sobreposição léxica de caracteres ou unigramas, a exemplo de *Exact Match* (EM) e F_1 -score lexical (PAPINENI et al., 2002; LIN, 2004). Contudo, essas abordagens evidenciam limitações em modelos generativos modernos (CALLISON-BURCH; OSBORNE; KOEHN, 2006; NOVIKOVA et al., 2017): saídas plenamente corretas, expressas por meio de paráfrases, estruturas oracionais elaboradas ou variações terminológicas válidas, são penalizadas caso divirjam da cadeia de caracteres estática estipulada no gabarito.

Para superar a rigidez léxica sem depender da morosidade de avaliações humanas em larga escala, Zheng et al. (2023) formalizaram o paradigma *LLM-as-a-Judge*, no qual modelos de linguagem de fronteira atuam como juízes semânticos automatizados. Para assegurar a validade interna do protocolo e mitigar vieses de avaliação, o processo ancora-se em três diretrizes metodológicas:

1. Cegamento procedimental duplo (*double-blind*): anonimização sistemática da estratégia geradora e embaralhamento pseudoaleatório das instâncias, prevenindo vieses atencionais de posição ou favorecimento de arquitetura;
2. Raciocínio prévio estruturado (*Few-Shot Chain-of-Thought*): obrigatoriedade de emissão de uma justificativa lógica detalhada, fundamentada nas sentenças de suporte oficiais, antes da consolidação do escore numérico final;

3. Triangulação cega multiavaliador: composição de um painel tripartite constituído por uma auditoria humana de referência e juízes sintéticos independentes de provedores distintos, mitigando o viés endógeno de autoavaliação (GU et al., 2024).

A reprodutibilidade e a consistência das notas emitidas pelo painel são avaliadas por meio do Coeficiente de Correlação Intraclassa (ICC), instrumento consolidado da teoria de mensuração (SHROUT; FLEISS, 1979; KOO; LI, 2016). Considerando um desenho de avaliação em que um conjunto idêntico de k juízes fixos audita todas as instâncias sob interesse nas médias dos avaliadores, emprega-se o modelo de efeitos mistos de duas vias com medidas médias ($ICC(3, k)$), computado analiticamente a partir do quadro de análise de variância (ANOVA) por:

$$ICC(3, k) = \frac{MS_R - MS_E}{MS_R}, \quad (2.4)$$

onde MS_R denota o quadrado médio entre os alvos avaliados e MS_E representa o quadrado médio do erro residual. Coeficientes iguais ou superiores a 0,75 indicam concordância inter-avaliadores substancial, atestando a confiabilidade métrica do painel (KOO; LI, 2016).

No âmbito da inferência estatística, a adoção convencional de testes de significância de hipótese nula baseados em valores- p (NHST) acarreta limitações documentadas na Engenharia de Software, como a sensibilidade desproporcional ao tamanho amostral e a incapacidade de estimar grandezas de efeito sob incerteza operacional (WASSERSTEIN; LAZAR, 2016; KITCHENHAM et al., 2017).

Diante disso, adota-se o paradigma da modelagem bayesiana com estimação de densidades posteriores completas, articulando o Intervalo de Mais Alta Densidade Posterior a 95% (HDI 95%) com Regiões de Equivalência Prática (ROPE) formalizadas *a priori* (KRUSCHKE, 2018; MCELREATH, 2020). Sob essa formulação, os contrastes pareados entre arquiteturas RAG são determinados examinando a densidade posterior relativa aos limites da ROPE:

1. Declara-se diferença prática substancial caso o HDI 95% recaia inteiramente fora do intervalo $[-ROPE, +ROPE]$;
2. Afirma-se equivalência prática operacional caso o HDI 95% esteja estritamente contido dentro dos limites da ROPE;
3. Caracteriza-se indefinição inferencial ou incerteza amostral caso o intervalo intercepte as fronteiras da ROPE ou o valor nulo, reportando-se a probabilidade posterior direta de superioridade $P(\theta > 0 \mid \mathcal{D})$ como métrica contínua de suporte à decisão de engenharia.

CAPÍTULO 3

REALIZAÇÃO DO EXPERIMENTO

Este capítulo detalha a metodologia experimental, o desenho do ambiente de software e os procedimentos de controle empregados no benchmark. Descrevem-se a arquitetura modular da esteira de automação, o protocolo de amostragem e congelamento histórico do corpus, as três estratégias de particionamento e recuperação avaliadas, as rotinas de telemetria de hardware e o protocolo cego de triangulação semântica por painel de juízes.

3.1 ARQUITETURA DO EXPERIMENTO E ESTRATÉGIAS RAG COMPARADAS

3.1.1 Visão Geral e Arquitetura Modular do Software

O ambiente experimental foi desenvolvido em linguagem Python (versão 3.11.9), estruturado sob uma arquitetura orientada à separação estrita de responsabilidades, visando garantir a reprodutibilidade científica, o isolamento dos ensaios e o controle de variáveis intervenientes.

O pipeline opera por meio de uma esteira determinística de dez módulos sequenciais (s01 a s10), articulados em quatro fases metodológicas independentes:

1. Preparação do corpus documental (s01 a s04): abrange a filtragem semântica dos registros do benchmark *HotpotQA*, a amostragem balanceada ($N = 180$), a extração dos artigos integrais da Wikipédia correspondentes e a conversão de formato com identificação estrutural via Docling;
2. Povoamento e indexação vetorial (s05): constrói as coleções isoladas no ChromaDB para cada grupo experimental e persiste a tabela relacional de mapeamento estrutural (*docstore*) em disco;
3. Execução pareada e telemetria (s06 e s07): realiza o ensaio piloto de calibração e a bateria principal de inferências serializadas sob concorrência unitária ($N = 1$), capturando simultaneamente a telemetria térmica e o consumo temporal nas fases físicas da GPU;

4. Consolidação analítica e auditoria (s08 a s10): agrega os registros brutos de telemetria em tabelas relacionais, gerencia o protocolo cego de triangulação inter-juízes (*LLM-as-a-Judge*) e executa as rotinas de verificação de integridade dos dados.

Para evitar duplicação algorítmica e assegurar consistência métrica entre os ensaios, a arquitetura é suportada por módulos transversais e um motor central de orquestração:

- **common.py**: provê as rotinas de infraestrutura e governança experimental, gerenciando o carregamento dos parâmetros em **config.yaml**, a persistência de logs de auditoria, a fixação de sementes pseudoaleatórias (**seed = 42**) e a interface com a base relacional SQLite local (**corpus.db**);
- **metrics.py**: centraliza as funções de normalização de texto e o cálculo determinístico das métricas primárias de recuperação e geração, abrangendo *Recall@k*, *Hit Rate*, *Exact Match*, *Containment Match* e *F₁-score* lexical;
- **engine.py**: orquestrador central de inferência, responsável por instanciar o modelo gerador local via Ollama (**llama3.1:8b-instruct-q4_K_M**), chavear o recuperador vetorial correspondente a cada estratégia e impor as restrições orçamentárias de contexto.

A Figura 3.1 ilustra a topologia física de diretórios e a organização dos módulos do projeto:

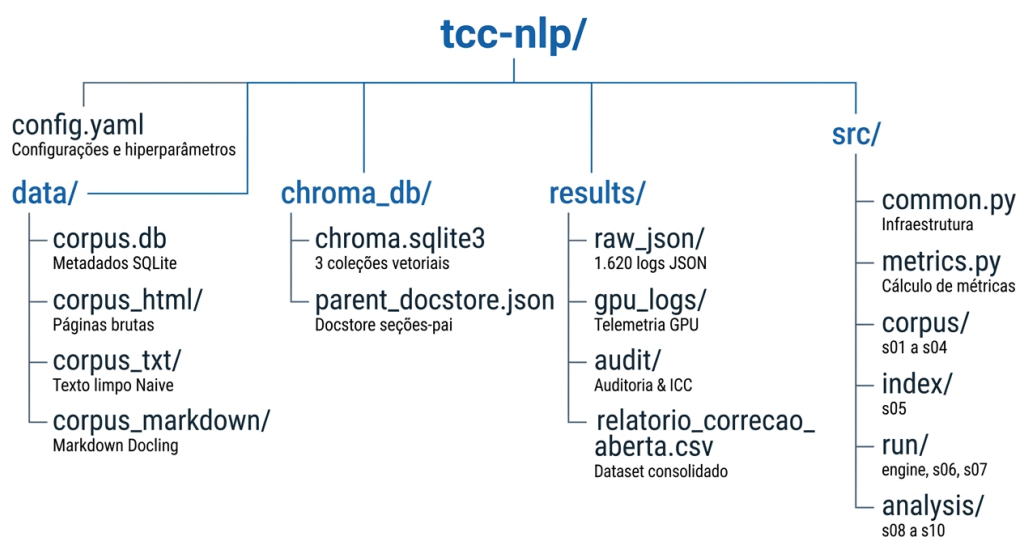


Figura 3.1: Estrutura de diretórios e organização modular do projeto.

Fonte: Autoras (2026).

3.1.2 Fluxo de Dados Ponta a Ponta

A esteira experimental estabelece um fluxo unidirecional, desacoplado e auditável, concebido para assegurar a rastreabilidade integral dos dados. Cada fase computacional consome entradas validadas e produz artefatos intermediários imutáveis persistidos em disco antes do avanço para a etapa subsequente, mitigando riscos de corrupção de estado em memória. Conforme ilustrado na Figura 3.2, o fluxo de dados desdobra-se em quatro fases operacionais:

1. Aquisição e estruturação do corpus: processa os registros brutos do benchmark *HotpotQA*, executa a amostragem estratificada por complexidade e realiza a extração automatizada dos artigos históricos na Wikipédia, convertendo-os em documentos estruturados via Docling e persistindo as tabelas de suporte na base relacional SQLite (`corpus.db`);
2. Indexação vetorial e persistência: gera as representações densas via *nomic-embed-text*, povoa de forma isolada as três coleções no ChromaDB e grava em disco o repositório de mapeamento hierárquico (*docstore*) com os metadados das seções-pai íntegras;
3. Inferência pareada e telemetria: submete as consultas ao modelo gerador local Llama 3.1 8B gerenciado pelo Ollama, sob execução pareada e serializada ($N = 1$), com permutação pseudoaleatória da ordem das estratégias por questão para neutralizar vieses residuais de aquecimento de cache e gravação síncrona da telemetria de hardware;
4. Consolidação métrica e triangulação avaliativa: consolida os registros brutos de telemetria em tabelas estruturadas, submete as respostas ao protocolo cego tripartite composto pela auditoria humana de referência e pelos modelos GPT-5 e Gemini 3.7 Pro, e computa os coeficientes psicométricos de confiabilidade inter-avaliadores via $ICC(3, 3)$.

3.1.3 Caracterização dos Cenários RAG e Desenho de Ablação

Para isolar o impacto de cada estratégia de engenharia de dados, o experimento estrutura as intervenções em três grupos de tratamento pareados. As representações vetoriais foram indexadas em coleções estanques no ChromaDB (`col_naive`, `col_structured` e `col_hierarchical`), impedindo qualquer contaminação cruzada durante as consultas. O fluxo comparativo de particionamento, indexação e resgate está ilustrado na Figura 3.3:

1. Grupo 1 (*Naive RAG*, linha de base): estabelece o controle clássico da literatura. Opera sobre o corpus em texto plano (`.txt`) submetido ao particionamento linear

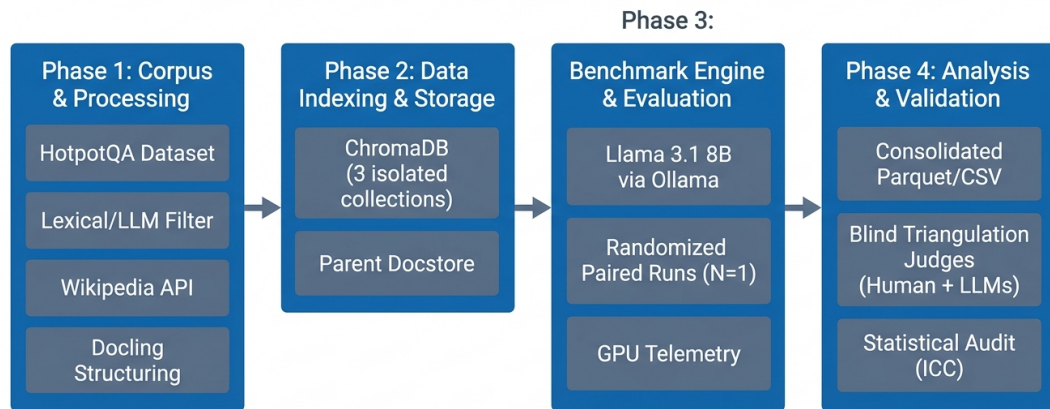


Figura 3.2: Fluxo de dados da esteira experimental e rastreabilidade dos artefatos intermediários.

Fonte: Autoria própria (2026).

via `SentenceSplitter`, fixado em blocos uniformes de 512 tokens com sobreposição estática de 50 tokens. A recuperação vetorial é realizada por similaridade de cosseno diretamente sobre os fragmentos indexados;

2. Grupo 2 (*Structured RAG*): isola o efeito da conversão estruturada e da segmentação orientada à sintaxe documental. Os documentos são transcritos em formato Markdown hierárquico via Docling, preservando a demarcação de títulos (`#`, `##`, `###`), listas e tabelas. O particionamento (`MarkdownNodeParser`) respeita as fronteiras semânticas das seções, aplicando subdivisão linear secundária apenas se uma seção exceder o teto operacional de 512 tokens. A etapa de busca vetorial permanece idêntica à do G1;
3. Grupo 3 (*Hierarchical RAG*): avalia o impacto adicional da recuperação multiescala em dois níveis (*parent-child*) sobre a base estruturada. Utiliza as seções íntegras do G2 como nós-pai (até 512 tokens), a partir das quais são derivados nós-filho menores de 128 tokens via `SentenceSplitter`. Apenas os nós-filho são incorporados à coleção vetorial. Durante a consulta, a rotina de resgate hierárquico substitui os nós-filho selecionados por suas respectivas seções-pai completas registradas no mapeamento relacional (`parent_docstore.json`), aplicando deduplicação determinística de identificadores para prevenir redundâncias no contexto.

O estudo configura um desenho fatorial incompleto orientado por contrastes lineares planejados, viabilizando a decomposição causal dos efeitos experimentais:

- Contraste estrutural ($G2 - G1$): mantendo constante a busca vetorial direta, mensura o efeito líquido da preservação do layout em Markdown e da segmentação delimitada por seções;
- Contraste de recuperação ($G3 - G2$): mantendo idêntica a base documental estruturada, quantifica o efeito aditivo da indexação de alta granularidade (128 tokens) combinada ao resgate determinístico de seções-pai íntegras.

Pela aditividade linear dos contrastes, o efeito global do benchmark ($G3 - G1$) desdobra-se na soma analítica $(G2 - G1) + (G3 - G2)$, permitindo atribuir os diferenciais observados em latência física, consumo de tokens e fidelidade gerativa ao respectivo estágio de intervenção. A validade interna do desenho repousa sobre dois controles estritos:

1. Paridade de profundidade e política de empacotamento: o parâmetro de busca vetorial foi fixado estritamente em $k = 20$ candidatos para os três grupos. A injeção no prompt segue uma política sequencial gulosa (*greedy packing*), preenchendo o contexto até o teto suportado pela janela de 8.192 tokens do Llama 3.1 8B;
2. Identidade de representação textual: as seções-pai resgatadas no Grupo 3 são rigorosamente idênticas aos blocos que compõem o corpus do Grupo 2, assegurando que o contraste $G3 - G2$ isole unicamente a mecânica de navegação topológica no espaço vetorial, sem divergência nas evidências textuais finais.

GRUPO 1: NAIVE RAG



GRUPO 2: STRUCTURED RAG



GRUPO 3: HIERARCHICAL RAG

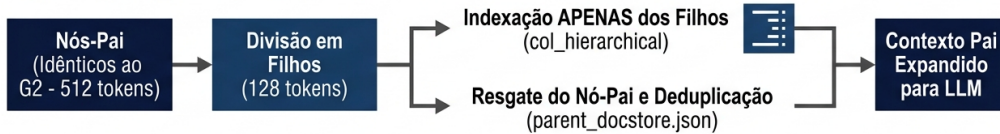


Figura 3.3: Diagrama arquitetural dos três cenários RAG avaliados.

Fonte: Autoria própria (2026).

3.1.4 Resolução Hierárquica (*parent_fetch*)

Na etapa de recuperação do Grupo 3 (Hierarchical RAG), adotou-se o protocolo determinístico *parent_fetch*. Diferentemente do algoritmo *AutoMergingRetriever* nativo do LlamaIndex (que exige o resgate da maioria dos nós-filho de uma mesma seção para disparar a promoção hierárquica), a rotina desenvolvida substitui de forma imediata cada nó-filho retornado no conjunto *top-k* pela sua respectiva seção-pai íntegra, indexada no arquivo *parent_docstore.json*. Essa decisão metodológica assegura o isolamento experimental e garante que qualquer evidência granular recupere seu contexto semântico integral antes da inferência.

Para prevenir a injeção redundante de texto no *prompt* e o desperdício do orçamento de contexto, o fluxo aplica deduplicação estrita por identificador: múltiplos nós-filho originários de uma mesma seção resultam na inclusão de uma única instância do nó-pai no contexto final entregue ao gerador. Para fins de ordenação, preserva-se o escore de similaridade por cosseno associado ao primeiro nó-filho resgatado daquela partição.

3.1.5 Gestão de Contexto e Orçamento Dinâmico de Tokens

A montagem do contexto para os três grupos segue um algoritmo estrito de orçamento dinâmico de tokens, substituindo heurísticas empíricas baseadas em contagem de caracteres. A capacidade disponível para injeção dos nós recuperados ($C_{\text{disponível}}$) é calculada deterministicamente por:

$$C_{\text{disponível}} = C_{\text{total}} - T_{\text{system}} - T_{\text{pergunta}} - T_{\text{predict}} - M_{\text{segurança}},$$

onde:

- $C_{\text{total}} = 8.192$ tokens representa a janela de contexto alocada no Ollama (*num_ctx*) para o LLaMA 3.1 8B;
- $T_{\text{predict}} = 512$ tokens é o teto máximo reservado para a geração da resposta (*num_predict*);
- $M_{\text{segurança}} = 64$ tokens absorve variações de formatação, delimitadores de *prompt* e separadores estruturais (*margem_contexto_tokens*);
- T_{system} e T_{pergunta} correspondem à extensão em tokens da instrução de sistema e da consulta avaliada, respectivamente.

Os parâmetros T_{system} e T_{pergunta} são pré-estimados em tempo de execução através de um *proxy* leve em Python utilizando a biblioteca *tiktoken* (codificação *cl100k_base*).

Como o servidor Ollama opera como um serviço de inferência desacoplado e não disponibiliza um *endpoint* síncrono de tokenização fracionária durante a montagem do texto, o `tiktoken` atua exclusivamente como uma estimativa rápida em microssegundos para guiar o fatiamento e prevenir o transbordamento do *buffer*. A margem de segurança $M_{\text{segurança}} = 64$ tokens foi calibrada especificamente para absorver divergências marginais de vocabulário entre o esquema `cl100k_base` (100k tokens) e o tokenizador BPE nativo do LLaMA 3.1 (128k tokens). Essa aproximação atua unicamente na esteira de montagem do *prompt*: toda a contagem oficial, canônica e auditada de tokens utilizada nos cálculos estatísticos, gráficos e tabelas deste estudo provém dos contadores nativos de telemetria da GPU devolvidos pela API do Ollama (`prompt_eval_count` para entrada e `eval_count` para geração).

Os fragmentos recuperados são alocados sequencialmente no *prompt* em ordem decrescente de similaridade de cosseno sob uma política de empacotamento guloso (*greedy packing*). Quando um fragmento excede o saldo orçamentário restante, é truncado proporcionalmente para ocupar a capacidade residual disponível (`nodes_partial`). Caso esse truncamento conservador ainda deixe uma folga marginal de tokens, o algoritmo continua a iterar sobre os nós subsequentes, fatiando frações adicionais até que o orçamento de contexto seja plenamente exaurido. Os fragmentos para os quais não resta saldo orçamentário são integralmente descartados (`nodes_dropped`). Esse mecanismo maximiza o preenchimento da janela de contexto sem violar o teto $C_{\text{disponível}}$, explicando a média de 2,15 nós parciais por requisição observada no *Naive RAG* (decorrente de fragmentos consecutivos cortados para absorver o resíduo orçamentário), prevenindo o transbordamento de VRAM e erradicando falhas de execução por estouro de contexto.

3.2 AMOSTRAGEM ESTRATIFICADA E SELEÇÃO DAS QUESTÕES

3.2.1 Fonte dos Dados e Recorte Temático

O conjunto de testes foi extraído do *benchmark* HotpotQA (Yang *et al.*, 2018) na configuração *distractor* (partições de treino e validação). A escolha fundamenta-se na exigência mandatária de raciocínio *multi-hop* (isto é, a necessidade de interligar trechos de documentos distintos para responder à pergunta), cenário em que a eficácia da recuperação de contexto dita diretamente o limite de desempenho do modelo gerador.

Para assegurar reprodutibilidade estrita, o carregamento da base fixa o *commit hash* no Hugging Face em `1908d6afbbead072334abe2965f91bd2709910ab`, prevenindo discrepâncias decorrentes de atualizações posteriores no repositório remoto.

O escopo experimental foi delimitado ao domínio de conhecimento geral e enciclopédico do HotpotQA (abrangendo história, geografia, ciências, biografias e cultura), atendendo a dois objetivos metodológicos:

1. **Diversidade semântica e representatividade:** submete a recuperação vetorial densa a uma variedade heterogênea de entidades e relações conceituais, reproduzindo a complexidade de cenários reais de produção;
2. **Predominância de prosa contínua:** prioriza artigos descritivos estruturados em seções, parágrafos, listas e tabelas, mitigando variáveis de confusão (como notações matemáticas avançadas ou código-fonte) que exigiriam *parsers* especializados.

3.2.2 Estratificação por Dificuldade e Sorteio Determinístico

A amostragem foi executada sobre as partições oficiais do HotpotQA, assegurando cobertura uniforme entre os níveis de complexidade cognitiva da tarefa. A amostra final do *benchmark* totalizou $N = 180$ questões, distribuídas equitativamente entre os três estratos nativos do conjunto:

- 60 questões fáceis (*Easy*);
- 60 questões médias (*Medium*);
- 60 questões difíceis (*Hard*).

O sorteio foi conduzido pelo *script* `src/corpus/s02_sample_stratified.py`, aplicando ordenação canônica por identificador e semente pseudoaleatória fixa (`seed = 42`). O algoritmo percorre a base preservando a proporção nativa entre as tipologias de raciocínio (*bridge* e *comparison*).

3.2.3 Validação Humana e Protocolo de Reposição 1:1

Após o sorteio computacional, a amostra passou por um protocolo de inspeção humana pré-experimental (cujos registros de conformidade foram consolidados em `data/questions/validacao_manual.csv`) com base em três critérios de higidez:

- **Verificabilidade factual:** garantia de que a formulação admite resposta objetiva e inequívoca;
- **Consistência do gabarito:** confirmação de que a resposta de referência reflete com fidelidade o texto original e o consenso histórico/científico;

- **Completude probatória:** checagem de que os fatos de suporte (*supporting facts*) anotados são suficientes para sustentar o encadeamento dedutivo.

Durante a triagem piloto (composta por 20 questões fáceis, 20 médias e 20 difíceis), dez instâncias que apresentavam ambiguidades irreconciliáveis foram substituídas pelo registro imediatamente subsequente do mesmo estrato, preservando a reposição estrita 1:1 e o determinismo do sorteio. A integridade de toda a amostra expandida ($N = 180$) foi auditada programaticamente quanto à presença de evidências, e a totalidade das 540 saídas finais geradas no experimento (180 questões submetidas às 3 arquiteturas) foi inspecionada manualmente no relatório de correção aberta.

3.3 CONSTRUÇÃO, SANITIZAÇÃO E AUDITORIA DO CORPUS

3.3.1 Resgate Histórico e Mitigação de *Data Drift*

Para compor a base de conhecimento sobre a qual as questões operam, foram identificados todos os títulos presentes nos campos de contexto (*supporting facts* e distratores) do dataset HotpotQA.

A ingestão direta da Wikipédia contemporânea introduziria um viés severo de desvio editorial (*editorial drift*). O benchmark HotpotQA foi construído sobre o *dump* oficial da Wikipédia em língua inglesa de 01 de outubro de 2017. Em testes preliminares com a versão ao vivo da enciclopédia, constatou-se que 75,3% (113 de 150) das sentenças de suporte deixaram de ser localizáveis como correspondências exatas no texto, devido a quase uma década de edições colaborativas.

Para neutralizar esse viés e garantir fidelidade histórica absoluta, implementou-se um protocolo de congelamento histórico no script `src/corpus/s03_download_articles.py`:

- A coleta consulta a MediaWiki Action API (`action=query&prop=revisions`) com ordenação reversa (`rvdir=older`) a partir do marco temporal `rvstart=2017-10-01T00:00:00Z`;
- O sistema identifica o identificador de revisão (`oldid`) e o *timestamp* da última versão publicada antes dessa data de corte;
- O HTML integral dessa revisão específica é baixado e congelado localmente em `data/raw/`, registrando-se os metadados no banco relacional `corpus.db`;
- Taxa de sucesso no download e inocuidade das falhas: do universo de 1.788 títulos únicos catalogados para a composição do corpus (dos quais 358 são artigos de suporte

e 1.430 são distratores), 1.780 foram efetivamente coletados e processados com êxito. As 8 únicas falhas registradas na MediaWiki Action API (`download_report.json`) decorreram da inexistência de revisões preservadas pré-2017 para páginas renomeadas ou eliminadas. A auditoria relacional no banco `corpus.db` atestou que 100% dos 8 artigos faltantes eram estritamente distratores (*ruído negativo de fundo*), registrando-se zero perdas entre os artigos de suporte. Assim, nenhuma evidência ou fato de suporte do gabarito oficial foi comprometido por falha de download.

3.3.2 Sanitização Prévia do DOM e Paridade de Formatos

Antes de gerar os arquivos para os três grupos, o HTML dos artigos passa por uma etapa de higienização para expurgar ruídos que distorceriam os *embeddings* e a verificação das respostas:

- Remoção de guias fonéticos e metadados inline: exclusão de elementos com a classe `<span class="noexcerpt">` (pronúncias em IPA e variações ortográficas) e caixas de navegação do rodapé (`.navbox`), alinhando o HTML ao extrator original do HotpotQA;
- Tratamento de hiperlinks e citações: supressão integral de notas de rodapé (`<sup>`) e desembrulho (*unwrap*) de tags de âncora (`<a>`), mantendo apenas o texto visível sem atributos `href` para evitar poluição lexical nos *embeddings*;
- Correção estrutural de abertura no Docling: como o *parser* HTML do Docling descarta textos anteriores ao primeiro cabeçalho, inseriu-se um cabeçalho sintético `<h2>Introdução</h2>` no início do `<body>`, assegurando a preservação integral do parágrafo introdutório no Markdown do G2 e G3.

A partir desse HTML limpo, a esteira gera os dois formatos de entrada: o arquivo `.txt` normalizado (G1), com supressão de tabelas e marcações, e o arquivo `.md` (G2 e G3), preservando a hierarquia sintática de seções, listas e tabelas.

3.3.3 Auditoria Automatizada e Limite Empírico de Cobertura

A integridade do corpus foi avaliada por meio de uma rotina de auditoria computacional em `src/corpus/s04_prepare_corpus.py`, registrada no artefato `results/audit/corpus_audit.json`:

- Auditoria automatizada por correspondência exata: a busca determinística por substring normalizada (Unicode NFKC, *casefold* e pontuação colapsada) localizou

deterministicamente 98,14% (421 de 429) dos fatos de suporte oficiais dentro dos arquivos gerados (.txt e .md);

- Conferência manual dos fatos residuais: a inspeção dos 8 fatos residuais confirmou que 100% das evidências necessárias estavam presentes nos artigos baixados, decorrendo a divergência estrita da busca automatizada de variações pontuais de espaçamento e abreviações herdadas do extrator original.

A auditoria confirmou que a representação dos fatos afeta de maneira idêntica os formatos em texto plano (G1) e estruturado (G2/G3), garantindo a validade interna dos testes.

A relação completa das 180 questões selecionadas, seus respectivos estratos de dificuldade (60 fáceis, 60 médias e 60 difíceis) e a lista dos 1.780 artigos que compõem o corpus encontram-se detalhadas nos Apêndices A e B, bem como no repositório de dados do projeto (data/questions/ e data/corpus.db).

3.4 AMBIENTE OPERACIONAL, INFRAESTRUTURA DE HARDWARE E SOFTWARE

3.4.1 Especificações de Hardware e Isolamento Experimental

Para assegurar reprodutibilidade estrita e blindagem contra oscilações de rede ou concorrência de processos no sistema operacional, toda a esteira experimental (Fases 1 a 4) foi executada em estação dedicada local, operando em regime de execução exclusivo ($N = 1$) e sem processos concorrentes em segundo plano:

- **Processador (CPU):** Intel Core i7-10700 (8 núcleos físicos / 16 *threads* lógicos, frequência base de 2,90 GHz com Turbo Boost até 4,80 GHz, 16 MB de Intel Smart Cache, arquitetura x86_64);
- **Unidade de Processamento Gráfico (GPU):** NVIDIA GeForce RTX 3050 (8 GB de VRAM GDDR6 dedicados, 2.560 núcleos CUDA, *driver* v596.36), mantendo alocação estável de ~4,7 GB de memória de vídeo durante a inferência do modelo LLaMA 3.1 8B (quantização Q4_K_M);
- **Memória Principal (RAM):** 32 GB DDR4 (frequência de operação em 2.933 MHz);
- **Armazenamento Secundário (HDD):** Disco rígido mecânico Western Digital Blue 1 TB (modelo WD10SPZX-21Z10T0), interface SATA III (6 Gb/s), velocidade de rotação de 5.400 RPM e 128 MB de *buffer* de memória cache;

- **Sistema Operacional:** Microsoft Windows 11 Home (64-bit, compilação 26200);
- **Ambiente de Execução e *Runtime*:** Python v3.11.9 gerenciado sob ambiente virtual isolado (`.venv`) e servidor de inferência local Ollama (v0.3+) com aceleração via CUDA.

A Tabela 3.1 sintetiza o ecossistema tecnológico e o papel metodológico de cada componente de *software* adotado no *benchmark*.

Tabela 3.1: Ecossistema de *software*, bibliotecas e componentes da arquitetura experimental.

Componente	Tecnologia	Versão / Configuração	Papel Metodológico no Experimento
Linguagem Base	Python	v3.11.9	Orquestração da esteira sequencial e <i>scripts</i> analíticos (<code>s01</code> a <code>s10</code>).
Servidor de Inferência	Ollama	<i>Daemon</i> v0.3+ / Cliente v0.33.2	Execução local com aceleração por GPU do LLaMA 3.1 8B (Q4_K_M) e <code>nommic-embed-text</code> .
Banco Vetorial	ChromaDB	v0.6.3 (persistência local)	Armazenamento de vetores em três coleções isoladas e busca por similaridade de cosseno.
<i>Parser</i> Estrutural	Docling (IBM)	v2.123.1 (<i>backend</i> HTML, OCR desligado)	Conversão de HTML sanitizado para Markdown preservando hierarquia de seções.
<i>Framework</i> RAG	LlamaIndex	v0.12.10 (<i>core</i>)	Primitivas de segmentação (<code>SentenceSplitter</code>), nós hierárquicos e estruturas de dados.
<i>Proxy</i> de Tokenização	Tiktoken	v0.8.0 (<i>encoding</i> <code>cl100k_base</code>)	Estimativa determinística de <i>tokens</i> para gestão do orçamento de contexto no <i>prompt</i> .
Banco Relacional	SQLite	<code>sqlite3</code> (<code>data/corpus.db</code>)	Armazenamento estruturado de metadados, amostras, artigos e rastreabilidade dos experimentos.
Modelagem Estatística	PyMC	PyMC v5.16+ e SciPy v1.13+	Ajuste bayesiano Student- <i>t</i> via NUTS e simulação direta de Monte Carlo.
Painel de Avaliação	LLMs e Humano	Auditora de Referência, GPT-5 e Gemini 3.7 Pro	Comitê cego tripartite para avaliação da acurácia semântica (<i>LLM-as-a-Judge</i>).

Fonte: Autoras (2026).

3.4.2 Modelos de Linguagem, Embeddings e Parametrização Técnica

O ecossistema experimental foi projetado para desacoplar a geração de inferência local (controle estrito de latência e hardware) da validação semântica das respostas:

- **Modelo gerador e classificador local (llama3.1:8b):**

- *Arquitetura*: LLaMA 3.1 8B Instruct (Meta AI), operando sob quantização de 4-bit (Q4_K_M) via Ollama local com aceleração CUDA;
- *Aplicações*: validação temática preliminar das questões (src/corpus/s01_filter_questions.py) e geração de respostas nas três estratégias RAG (src/run/engine.py e s07_benchmark.py);
- *Temperatura*: configurada estritamente em 0.0 (decodificação determinística / greedy decoding, eliminando estocasticidade na geração);
- *Janela de contexto (num_ctx)*: 8.192 tokens na inferência RAG (2.048 tokens no classificador temático preliminar);
- *Teto de resposta (num_predict)*: limite máximo de 512 tokens;
- *Persistência de VRAM*: parâmetro keep_alive: -1, mantendo os tensores do modelo continuamente alocados na memória da GPU (~4,7 GB), eliminando variações de latência por recarga de pesos entre requisições.

- **Modelo de representação vetorial (nomic-embed-text):**

- *Arquitetura*: Nomic Embed Text v1.5 (Nomic AI), instanciado localmente via Ollama;
- *Aplicações*: indexação vetorial dos fragmentos textuais (src/index/s05_index_all.py) e codificação em tempo de execução das consultas de busca;
- *Janela de contexto vetorial*: parametrizada em 4.096 tokens (embedding_num_ctx: 4096), acomodando a quase totalidade das seções estruturadas (com truncamento residual restrito a apenas 35 nós atípicos no acervo: 22 no Structured RAG e 13 no Hierarchical RAG, auditados em results/audit/nos_truncados_col_*.json);
- *Dimensionalidade e métrica*: vetores densos de 768 dimensões com indexação sob métrica de Similaridade por Cosseno no ChromaDB.

3.5 PROTOCOLO DE EXECUÇÃO PAREADA, ALEATORIZAÇÃO E CONTROLE DE VIÉS

3.5.1 Desenho Experimental Pareado e Volume Amostral

A execução da bateria de testes pelo script `src/run/s07_benchmark.py` adota um desenho experimental em blocos completos pareados (*repeated measures*). As 180 questões selecionadas foram submetidas uniformemente aos três cenários RAG (G1, G2 e G3) ao longo de 3 repetições independentes ($R = 3$):

$$N_{\text{total}} = 180 \text{ questões} \times 3 \text{ tratamentos} \times 3 \text{ repetições} = 1.620 \text{ ensaios de inferência}$$

Essa estrutura pareada totaliza 540 pares únicos (Questão $\times$ Cenário RAG) e garante que cada pergunta atue como seu próprio controle de variabilidade, permitindo isolar o efeito específico do tratamento experimental e absorver a heterogeneidade intrínseca de complexidade das questões na modelagem estatística posterior.

3.5.2 Aleatorização de Ordem e Rodadas Globais Intercaladas

Para neutralizar o viés de ordem de execução (*sequence bias*), efeitos cumulativos de aquecimento térmico da GPU e eventuais artefatos de memória *cache* do banco vetorial ChromaDB ou do servidor Ollama, o plano de execução foi estruturado pela função `montar_plano()` (`src/run/engine.py`) sob semente pseudoaleatória fixa (`seed = 42`):

- Rodadas globais intercaladas: as 3 repetições foram executadas em blocos sequenciais macro (Rodadas 1, 2 e 3), de modo que uma repetição completa dos 540 ensaios ocorresse antes do início da rodada seguinte;
- Emparelhamento com aleatorização intra-questão: dentro de cada rodada, a sequência das 180 questões foi aleatorizada deterministicamente via `rng.shuffle()`. Para cada questão, os três cenários RAG (G1, G2 e G3) foram executados consecutivamente sob uma ordem previamente sorteada para aquela pergunta (por exemplo, Questão A: $G2 \rightarrow G1 \rightarrow G3$; Questão B: $G3 \rightarrow G2 \rightarrow G1$). Essa abordagem garante que os três tratamentos de uma mesma pergunta sejam testados sob condições operacionais e térmicas de hardware estritamente homogêneas, enquanto a rotação pseudoaleatória dos cenários neutraliza o viés de precedência e previne artefatos residuais de cache de prefixo.

3.5.3 Regime Estrito de Concorrência Serial ($N = 1$)

A orquestração do motor RAG operou sob regime estritamente serializado ($N = 1$):

- Uma nova inferência só é disparada após a finalização completa da resposta anterior, a persistência atômica do artefato JSON intermediário em disco (diretório `results/raw_json/`) e a liberação das estruturas de memória volátil;
- Esse isolamento impede que requisições paralelas disputem largura de banda do barramento PCI Express, núcleos CUDA ou capacidade de VRAM na GPU, assegurando que as métricas de latência e consumo de recursos reflitam com fidelidade o custo computacional isolado de cada estratégia.

3.5.4 Eliminação do Efeito de Carga Inicial (*Warm-up Protocol*)

Antes de iniciar a contagem oficial dos 1.620 ensaios de inferência, o script executa um protocolo de aquecimento (*warm-up*) submetendo 3 perguntas enciclopédicas fora da amostra oficial aos modelos (`aquecer()` em `engine.py`):

- Essas 3 chamadas preliminares forçam o servidor Ollama a carregar previamente os pesos do modelo `llama3.1:8b` e do embedder `nomic-embed-text` na VRAM da GPU, alocando as estruturas de contexto;
- Os resultados e latências desse aquecimento são descartados por completo, eliminando o pico artificial de latência de inicialização (*cold start latency*) da primeira execução do benchmark.

3.5.5 Telemetria Contínua de Infraestrutura em Segundo Plano

Durante toda a execução dos 1.620 ensaios, uma thread assíncrona dedicada (`MonitorGPU`) coletou continuamente o estado físico do hardware a intervalos fixos de 10 segundos, gravando os registros em log CSV estruturado (`results/gpu_logs/gpu_<timestamp>.csv`):

- Unidade Gráfica (GPU): temperatura operacional do encapsulamento (°C), alocação absoluta de VRAM (MB) e percentual de utilização dos núcleos de computação via `nvidia-smi`;
- Memória do Sistema Host: consumo de RAM física (MB) e uso de memória virtual de paginação / SWAP (MB) via biblioteca `psutil`;

- Verificação de Estabilidade: o monitoramento confirmou a ausência de estrangulamento térmico (*thermal throttling*) e atestou que a alocação de VRAM permaneceu estável dentro do teto de ~4,7 GB, operando com consumo de paginação em disco (*swap*) estritamente nulo durante todo o experimento.

3.6 INDICADORES DE DESEMPENHO, MÉTRICAS DE TELEMETRIA E MAPEAMENTO DE DADOS

3.6.1 Métricas de Eficácia da Recuperação de Contexto

Para mensurar a eficácia dos módulos de busca vetorial e a integridade da evidência factual injetada no *prompt*, quantificou-se o resgate das sentenças de suporte (*supporting facts*) originais do HotpotQA (Yang *et al.*, 2018):

- **Recall@*k* de Evidências (`recall_at_k`):** proporção de fatos de suporte necessários (S_i) que foram efetivamente resgatados no conjunto de fragmentos recuperados (R_i):

$$\text{Recall@}k = \frac{|S_i \cap R_i|}{|S_i|};$$

- **Taxa de Resgate Integral (*Strict Hit Rate* / `hit_rate`):** indicador binário que atesta se o mecanismo de recuperação capturou a totalidade (100%) das evidências necessárias para a resolução da questão *multi-hop*:

$$\text{Hit Rate} = \begin{cases} 1, & \text{se } |S_i \cap R_i| = |S_i|, \\ 0, & \text{se } |S_i \cap R_i| < |S_i|; \end{cases}$$

- **Auditoria de Evidências e Mitigação de Falsos Negativos:** embora o cômputo automatizado utilize um algoritmo duplo de correspondência (`fato_presente()` em `src/metrics.py`, combinando correspondência por subcadeia normalizada e sobreposição léxica com limiar $\geq 75\%$), a integridade das evidências foi submetida à auditoria humana na coluna `validacao_manual_evidencias` do conjunto mestre (`results/relatorio_correcao_aberta.csv`), assegurando que variações tipográficas residuais não gerassem falsos negativos.

3.6.2 Métricas Lexicais Secundárias de Qualidade da Resposta (Análise Descritiva)

A avaliação das respostas em relação ao gabarito de referência (*ground truth*) foi calculada no módulo `src/metrics.py` sob normalização prévia de texto (conversão para

caixa baixa e supressão de pontuação, artigos e caracteres especiais). Essas métricas foram mantidas em caráter estritamente descritivo, garantindo comparabilidade com a literatura clássica de processamento de linguagem natural:

- **Correspondência Contida (*Containment Match* / `containment_match`):** indicador binário (1 ou 0) que afere se a totalidade dos termos constitutivos do gabarito oficial está contida no conjunto de *tokens* da resposta gerada (`all(token in pred_tokens for token in gold_tokens)`). O indicador atenua a rigidez do *Exact Match* quando o modelo emite uma sentença completa e correta contendo a entidade-alvo;
- **Correspondência Exata (*Exact Match*, `EM` / `exact_match`):** métrica estrita que exige igualdade absoluta (1,0) entre a cadeia predita normalizada e a resposta de referência;
- **F_1 -Score Lexical (`f1_score`):** média harmônica entre a precisão (P_{lex}) e a revocação (R_{lex}) no nível de unigramas compartilhados entre a resposta gerada e o gabarito:

$$F_1 = 2 \times \frac{P_{\text{lex}} \times R_{\text{lex}}}{P_{\text{lex}} + R_{\text{lex}}}.$$

Nota Metodológica sobre o Papel do F_1 : Em virtude da penalização sistemática que o F_1 impõe a sentenças explicativas corretas e paráfrases emitidas por modelos autoregressivos, o indicador não foi adotado como variável dependente na modelagem inferencial confirmatória de H_2 , cabendo esse papel exclusivamente à Acurácia Semântica Triangulada definida na Seção 3.7.

3.6.3 Decomposição Temporal e Telemetria de Latência

A latência de execução foi registrada com resolução de milissegundos, decompondo o ciclo de vida de cada requisição em quatro componentes temporais extraídos do motor de execução (`src/run/engine.py`) e da telemetria do Ollama:

- **Tempo de Recuperação (T_{rec} / `tempo_recuperacao_sec`):** intervalo consumido na codificação vetorial da consulta e na busca por cosseno no ChromaDB (incluindo a resolução e fusão de nós-pai no Grupo 3);
- **Tempo de Ingestão de Contexto (T_{prefill} / `tempo_prefill_sec`):** duração do processamento e ingestão dos *tokens* do *prompt* compilado na GPU pelo modelo gerador (`prompt_eval_duration`);
- **Tempo de Geração (T_{gen} / `tempo_geracao_sec`):** duração da decodificação e amostragem autorregressiva *token a token* da resposta (`eval_duration`);

- **Latência Total** (T_{total} / `tempo_total_sec`): tempo decorrido de ponta a ponta (*end-to-end*), desde a emissão da consulta até a decodificação do último *token* de saída.

3.6.4 Alocação de Contexto e Métricas de Custo Computacional

Para mensurar o custo computacional e a taxa de ocupação da janela de contexto de 8.192 *tokens*, foram monitoradas as variáveis de consumo e particionamento:

- **Tokens Oficiais de Entrada** (`tokens_prompt` / `prompt_eval_count`): contagem canônica de *tokens* do *prompt* (instrução do sistema, fragmentos recuperados e consulta) processados na GPU, obtida via telemetria nativa da API do Ollama (distinta da estimativa preliminar de montagem via `tiktoken`);
- **Tokens Oficiais de Saída** (`tokens_resposta` / `eval_count`): volume total de *tokens* emitidos na resposta decodificada pelo LLaMA 3.1 8B;
- **Auditoria de Fragmentos do Orçamento Dinâmico:**
 - `blocos_retornados` (`nodes_returned`): total de nós devolvidos da busca vetorial (*top-k* configurado);
 - `blocos_usados` (`nodes_used`): quantidade de fragmentos inseridos no *prompt* (sejam íntegros ou parciais);
 - `blocos_parciais` (`nodes_partial`): quantidade de fragmentos submetidos a truncamento proporcional para absorver o resíduo marginal da capacidade orçamentária ($C_{\text{disponível}}$);
 - `blocos_descartados` (`nodes_dropped`): volume de nós resgatados que foram integralmente excluídos por esgotamento do teto de contexto.

3.6.5 Consolidação do Dataset Mestre e Estrutura de Artefatos

Os dados brutos, telemetrias e saídas textuais dos 1.620 ensaios de inferência (consolidados nos 540 pares únicos submetidos ao julgamento semântico) foram unificados no arquivo tabular `results/relatorio_correcao_aberta.csv`. O conjunto é indexado pelo identificador sequencial `numero_execucao` ($1 \leq i \leq 540$) e distribui-se em cinco dimensões de análise:

1. **Identificação Experimental:** `numero_execucao`, arquitetura (G1, G2 ou G3), dificuldade (*easy*, *medium*, *hard*) e `id_pergunta`;

2. **Telemetria de Tempo e Recursos:** `tempo_recuperacao_sec`, `tempo_prefill_sec`, `tempo_geracao_sec`, `tempo_total_sec`, `tokens_prompt`, `tokens_resposta` e `tokens_total`;
3. **Orçamento de Contexto:** `blocos_retornados`, `blocos_usados`, `blocos_parciais` e `blocos_descartados`;
4. **Desempenho de Recuperação e Geração:** `fatos_recuperados`, `total_fatos`, `recall_at_k`, `hit_rate`, `containment_match`, `exact_match`, `f1_score`, `pergunta`, `gabarito_resposta`, `resposta_gerada`, `gabarito_evidencias` e `evidencias_recuperadas`;
5. **Metadados de Validação e Julgamento:** `validacao_manual_resposta`, `validacao_manual_evidencias` e `observacoes_humanas`.

3.7 PROTOCOLO DE AVALIAÇÃO CEGA E VALIDAÇÃO DA QUALIDADE SEMÂNTICA

3.7.1 Superação das Limitações Lexicais via Julgamento Semântico e Definição de H_2

Métricas baseadas em sobreposição de cadeias de caracteres (como *Exact Match* e *F1-Score*) penalizam de maneira desproporcional respostas que formulam o conceito correto por meio de paráfrases, sinônimos ou sentenças completas. Enquanto em tarefas extrativas telegráficas o F_1 opera de forma adequada, em tarefas de geração aberta com raciocínio dedutivo, respostas factualmente exatas (e.g., “*O diretor do filme é Christopher Nolan*”) recebem pontuações de F_1 artificialmente deprimidas quando comparadas a um gabarito restrito (“*Christopher Nolan*”).

Por essa razão metodológica, adotou-se a Acurácia Semântica Triangulada (Nota Média $\geq 8,0$) como a variável dependente formal primária para a validação das hipóteses H_{2a} e H_{2b} (modelada bayesianamente por regressão logística hierárquica), restringindo o F_1 -score e o *Exact Match* à função de indicadores léxicos secundários.

3.7.2 Protocolo de Cegamento e Anonimização

A avaliação foi conduzida sob duplo-cegamento (*double-blind*) implementado no *script* `src/analysis/s09_judge_protocol.py`:

- **Anonimização por Identificador Cego (`blind_id`):** o identificador da arquitetura (G1, G2 ou G3) foi removido de cada registro e substituído por um *hash* criptográfico

SHA-1 opaco de 12 caracteres;

- **Embaralhamento Global:** a ordem dos pares avaliados foi totalmente pseudoaleatorizada antes da entrega aos julgadores, impedindo a identificação da arquitetura por correlação de sequência ou bloco;
- **Omissão de Metadados de Origem:** os avaliadores tiveram acesso unicamente aos quatro campos indispensáveis à auditoria factual: Pergunta, Sentenças de Suporte oficiais, Resposta de Referência e Resposta Gerada pelo modelo;
- **Amostragem para Julgamento e Estabilidade de Decodificação:** enquanto as análises de latência e telemetria utilizaram a mediana de três repetições técnicas completas ($R = 3$), submeteu-se ao painel de julgamento a resposta gerada na primeira execução técnica (`repeticao == 1`) de cada par experimental ($N = 540$ respostas). Em virtude da fixação de semente (`seed = 42`) e temperatura nula ($T = 0, 0$), a estabilidade textual foi de 99,63% (com 538 dos 540 pares emitindo saídas estritamente idênticas caractere a caractere entre as rodadas). Nos dois únicos casos de divergência marginal (decorrentes de não-determinismo de ponto flutuante em operações paralelas na GPU), preservou-se formalmente o texto emitido no primeiro ensaio.

3.7.3 Painel de Triangulação de Três Juízes

Com o intuito de aliar rigor substantivo, escalabilidade e independência, estabeleceu-se um comitê tripartite composto por uma auditoria humana de referência e dois modelos de linguagem de fronteira de fornecedores distintos:

- **Auditora Humana (Referência Primária):** inspeção conduzida por uma das autoras sob cegamento absoluto dos identificadores de arquitetura e aleatorização dos 540 registros (`notas_humano.csv`), avaliando a equivalência conceitual frente ao gabarito e às sentenças de suporte;
- **Juiz A (GPT-5, OpenAI):** avaliador algorítmico da família GPT, consultado via protocolo estrito com encadeamento de raciocínio (*Few-Shot Chain-of-Thought*), temperatura determinística ($T = 0, 0$) e rubrica contínua padronizada;
- **Juiz B (Gemini 3.7 Pro, Google AI):** avaliador algorítmico da família Gemini, empregado para mitigar viés de fornecedor único e arquitetura, sob idênticos parâmetros e rubrica de avaliação.

Os sistemas de IA operaram sob decodificação determinística ($T = 0, 0$) e receberam a mesma rubrica de pontuação de 0 a 10 pontos acompanhada de âncoras conceituais

(*Few-Shot CoT*), discriminando respostas factualmente idênticas, parcialmente corretas ou incorretas/alucinadas.

3.7.4 Concordância Interavaliadores e Salvaguarda Metodológica

A confiabilidade entre as notas atribuídas pelos três juízes foi avaliada por meio do Coeficiente de Correlação Intraclassa, $ICC(3, k)$ (modelo de efeitos mistos de duas vias com média de avaliadores fixos) (Shrout; Fleiss, 1979):

$$ICC(3, k) = \frac{MS_R - MS_E}{MS_R},$$

onde MS_R denota o quadrado médio entre os itens avaliados e MS_E representa o quadrado médio do erro residual.

- **Limiar de Validação Concorrente:** estipulou-se *a priori* o patamar de corte em $ICC \geq 0,75$, correspondente ao nível de concordância excelente;
- **Salvaguarda Metodológica Estrita:** definiu-se formalmente que, caso o $ICC(3, k)$ ficasse aquém de $0,75$, as avaliações dos modelos de linguagem seriam desqualificadas, recorrendo-se exclusivamente à auditoria manual humana como o padrão-ouro definitivo nas análises inferenciais.

3.7.5 Modelagem Inferencial Bayesiana e Validação das Hipóteses

A análise das oito hipóteses apoiou-se em inferência bayesiana pareada, motivada pela assimetria dos dados empíricos e pela capacidade de estimar diretamente as distribuições de probabilidade dos contrastes (McElreath, 2020). As especificações estatísticas respeitaram a estrutura de cada desfecho:

- **Latência Total e Fases Componentes (H_3):** contrastes pareados modelados por regressão linear robusta baseada na distribuição Student- t via biblioteca *PyMC* (Abril-Pla *et al.*, 2023), utilizando o amostrador NUTS com quatro cadeias de Monte Carlo, 2.000 iterações de amostragem e 1.000 de aquecimento (*warm-up*), obtendo convergência plena ($\hat{R} = 1,000$ e $ESS > 5.000$);
- **Consumo de Tokens de Entrada (H_4):** distribuição a posteriori derivada por amostragem direta de Monte Carlo sob distribuição Student- t com *prior* de Jeffreys (20.000 iterações; intervalos a 95% via percentis 2,5% e 97,5%). A solução analítica direta dispensou aproximação iterativa por MCMC em virtude de os *tokens* resultarem de regras algorítmicas determinísticas de empacotamento;

- **Eficácia de Recuperação (H_1):** modelo bayesiano pareado das diferenças de Recall@ k entre estratégias por questão;
- **Variáveis Binárias de Sucesso (H_1 e H_2):** o *Hit Rate* e a Acurácia Semântica Triangulada (nota média $\geq 8,0$) foram modelados via regressão logística pareada com verossimilhança binomial, extraindo a diferença de proporções (Δp), a razão de chances (*Odds Ratio*, OR) e o HDI de 95%.

A inferência confirmatória articulou o Intervalo de Mais Alta Densidade Posterior a 95% (HDI 95%) com Regiões de Equivalência Prática (ROPE) pré-fixadas (Kruschke, 2018):

1. **Faixa ROPE para Latência:** delimitada em $\pm 10\%$ da média do grupo de controle ($-0,840$ s a $+0,840$ s), absorvendo oscilações operacionais térmicas e do sistema operacional;
2. **Faixa ROPE para Tokens:** estipulada em ± 200 tokens ($\sim 2,6\%$ da média do *Naive RAG*), patamar abaixo do qual flutuações residuais são inexpressivas para custos de produção;
3. **Critério Decisório sob ROPE:** a superioridade prática exigiu a exclusão integral do HDI 95% dos limites da ROPE na direção da hipótese alternativa ($P = 100,0\%$), enquanto a equivalência prática demandou o confinamento estrito de 100% do intervalo no interior da faixa;
4. **Métricas sem Limiar ROPE (Recall@ k e Acurácia Semântica):** a análise fundamentou-se no contraste posterior (Δ), na razão de chances e na probabilidade de superioridade $P(\Delta > 0)$. Intervalos credíveis que interceptam o valor nulo caracterizam a não rejeição da hipótese de equivalência (H_0). Os contrastes primários avaliaram o efeito da estruturação (G1 vs. G2) e o efeito incremental da recuperação hierárquica (G2 vs. G3), reportando o contraste entre G3 e G1 como estimativa derivada do modelo.

3.8 REPRODUTIBILIDADE, AUDITORIA E ASPECTOS ÉTICOS

Para assegurar a auditabilidade integral dos experimentos e o pleno alinhamento às práticas de Ciência Aberta, a esteira experimental foi estruturada sobre três pilares de governança técnica e metodológica:

1. **Reprodutibilidade e congelamento de parâmetros:** todos os hiperparâmetros, sementes pseudoaleatórias (`seed = 42`), versões de dependências, *prompts* estáticos (temperatura 0,0) e pesos dos modelos locais (`llama3.1:8b-instruct-q4_K_M` e `nomic-embed-text`) encontram-se centralizados no arquivo de configuração `config.yaml`.
2. **Trilha de auditoria e dados abertos:** todas as etapas da esteira geram registros persistentes no diretório `results/`, incluindo os relatórios de triagem e resgate enciclopédico (`classificacoes.jsonl`, `download_report.json`), a validação automatizada de evidências (`corpus_audit.json`), os julgamentos do painel tripartite (`judges_report.json`), a telemetria física da GPU (`gpu_logs/`, `raw_json/`) e a matriz consolidada de dados (`relatorio_correcao_aberta.csv`).
3. **Conformidade ética e soberania de dados:** o estudo utilizou exclusivamente bases públicas e sob licenciamento livre (o *benchmark* HotpotQA, sob licença CC BY-SA 4.0, e a Wikipédia, sob licença CC BY-SA 3.0), sem coleta, processamento ou armazenamento de dados sensíveis ou informações de identificação pessoal (*Personally Identifiable Information*, PII). Toda a esteira principal de recuperação vetorial, injeção de contexto e inferência generativa operou de forma 100% local via servidor Ollama em ambiente isolado, ao passo que as chamadas às APIs externas na fase de validação semântica receberam exclusivamente textos anonimizados desprovidos de metadados experimentais.

3.8.1 Declaração de Uso de Ferramentas de IA na Implementação Computacional

A infraestrutura computacional e os módulos de *software* empregados neste estudo (compreendendo a esteira de ingestão documental com Docling, persistência vetorial no ChromaDB, orquestração de inferência local via Ollama/LLaMA 3.1, além dos *scripts* de telemetria de *hardware* e modelagem bayesiana) contaram com suporte de síntese assistida por modelos de linguagem de grande escala (como Claude e Gemini), a partir de engenharia de *prompts* e especificações arquiteturais detalhadas elaboradas pelas autoras.

Todos os códigos gerados foram submetidos a testes empíricos rigorosos, checagem estática de tipos, auditoria lógica e monitoramento de execução em ambiente local. A concepção metodológica, o desenho experimental, a formulação das hipóteses, o controle de variáveis de *hardware* e a interpretação científica dos resultados são de autoria e responsabilidade exclusivas das autoras.

RESULTADOS E DISCUSSÃO

4.1 ANÁLISE COMPARATIVA DA LATÊNCIA DE RESPOSTA E EFICIÊNCIA DE HARDWARE

A latência total de resposta constitui um dos principais gargalos de viabilidade operacional na implantação de arquiteturas RAG em ambientes de produção (Gao *et al.*, 2023). No presente estudo, a latência de cada inferência foi capturada de forma isolada, em concorrência serializada ($N = 1$) com controle estrito de temperatura da GPU e monitoramento de recursos locais, conforme detalhado na Seção 3.4.

A análise foi conduzida sobre a mediana das repetições pareadas para as 180 questões do *benchmark* (540 ensaios consolidados), considerando tanto o tempo total de inferência quanto a sua decomposição granular nas três etapas canônicas do fluxo: (i) busca vetorial no banco de dados (*Retrieval*), (ii) processamento do contexto e preenchimento da memória de atenção (*Prefill*) e (iii) decodificação autorregressiva da resposta (*Generation*), conforme sintetizado na Figura 4.1.

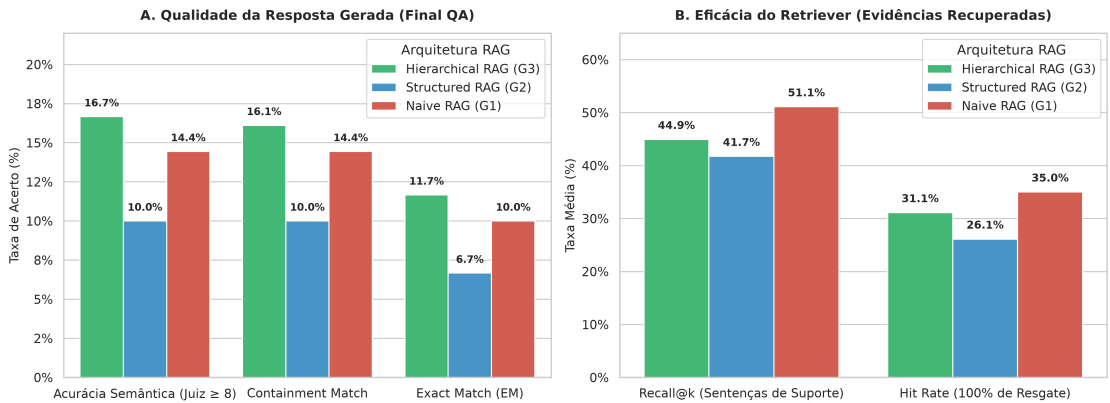


Figura 4.1: Resumo comparativo do desempenho global entre os cenários RAG.

Fonte: Autoras (2026).

4.1.1 Desempenho Global da Latência: Comparativo entre Arquiteturas

Os resultados descritivos do tempo total de execução revelam uma disparidade substancial entre a abordagem tradicional (*Naive RAG*) e as abordagens que utilizam estruturação prévia de documentos (*Structured* e *Hierarchical RAG*). A Tabela 4.1 sintetiza as estatísticas descritivas observadas para as 180 questões do *corpus*.

Tabela 4.1: Estatísticas descritivas da latência total de inferência por arquitetura RAG ($N = 180$).

Arquitetura RAG	Média (s)	Desvio Padrão (s)	Mediana (s)	Intervalo Interquartil IQR (s)	Mínimo (s)	Máximo (s)
Hierarchical RAG (G3)	6,303	0,917	6,311	1,167	3,763	8,247
Structured RAG (G2)	6,278	1,004	6,223	1,442	3,678	8,365
Naive RAG (G1, Controle)	8,403	0,195	8,377	0,082	8,112	8,834

Fonte: Autoras (2026).

Conforme observado na Tabela 4.1, o grupo de controle (*Naive RAG*) registrou uma latência média de 8,403 s, operando em um regime sensivelmente mais lento do que as arquiteturas estruturadas. Notavelmente, o grupo manifestou baixíssima dispersão temporal, com desvio-padrão de apenas 0,195 s e intervalo interquartil de 0,082 s, comportamento derivado da saturação uniforme do orçamento de contexto na etapa de *prefill*. Em contrapartida, o *Hierarchical RAG* e o *Structured RAG* apresentaram médias de 6,303 s e 6,278 s, respectivamente, alcançando uma redução relativa de aproximadamente 25,0% no tempo total de resposta, conforme ilustrado no perfil de distribuição empírica da Figura 4.2.

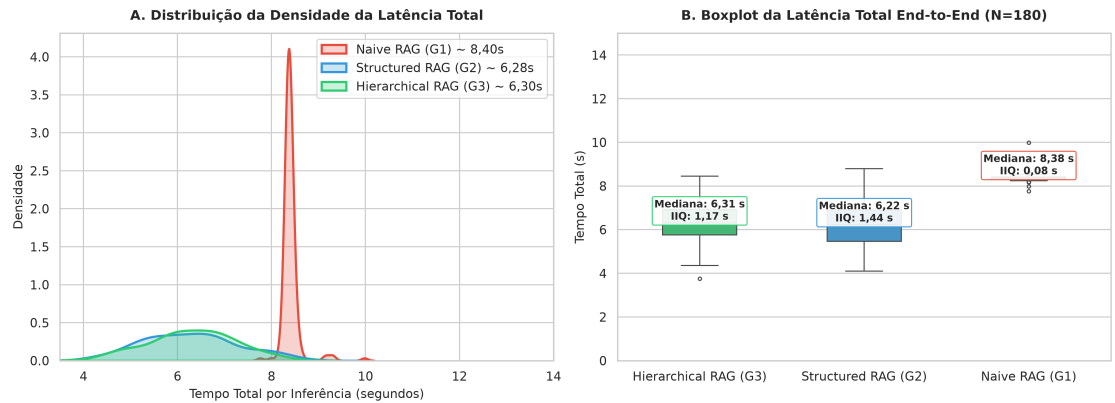


Figura 4.2: Distribuição empírica da latência total por arquitetura RAG.

Fonte: Autoras (2026).

4.1.2 Modelagem Bayesiana da Latência Pareada

Para avaliar o efeito das arquiteturas sob uma perspectiva probabilística robusta e imune a eventuais assimetrias distributivas, implementou-se uma regressão baseada na distribuição Student-*t* pareada por questão via biblioteca *PyMC* (Abril-Pla *et al.*, 2023;

Kruschke, 2015). A modelagem estimou simultaneamente a distribuição a posteriori das diferenças pareadas (Δ) entre os três cenários experimentais.

Definiu-se uma Região de Equivalência Prática (ROPE) de $\pm 10\%$ em relação à média do grupo controle (Kruschke, 2018), delimitada no intervalo $[-0,840 \text{ s}; +0,840 \text{ s}]$. Diferenças cujo Intervalo de Mais Alta Densidade Posterior (HDI 95%) repousem integralmente fora do ROPE atestam ganhos não apenas credíveis do ponto de vista estatístico, mas dotados de relevância prática substantiva. A Tabela 4.2 consolida os parâmetros a posteriori estimados pelas cadeias MCMC (quatro cadeias, 2.000 *draws*, 1.000 iterações de *warm-up*, registrando convergência plena com $\hat{R} = 1,000$ e $\text{ESS} > 5.000$).

Tabela 4.2: Estimativas a posteriori das diferenças pareadas de latência (Δ Tempo em segundos).

Comparação Pareada	Média Posterior $\Delta\mu$ (s)	Desvio Padrão	HDI 95% Inferior (s)	HDI 95% Superior (s)	% no ROPE	$P(\text{Superioridade})$
Hierarchical vs. Naive (G3 – G1)	-2,097	0,070	-2,230	-1,956	0,0%	100,0%
Structured vs. Naive (G2 – G1)	-2,132	0,074	-2,273	-1,986	0,0%	100,0%
Hierarchical vs. Structured (G3 – G2)	+0,035	0,101	-0,162	+0,231	100,0%	0,0%

Nota: Para Hierarchical vs. Structured, a probabilidade indica que a diferença repousa 100% contida no interior da faixa de equivalência prática da ROPE. Fonte: Autoras (2026).

A análise a posteriori confirma de forma categórica a hipótese de superioridade em eficiência temporal das abordagens estruturadas:

1. **Hierarchical vs. Naive:** a diferença média estimada foi de $-2,097 \text{ s}$ (HDI 95%: $[-2,230 \text{ s}; -1,956 \text{ s}]$), situando-se inteiramente à esquerda da fronteira inferior da ROPE ($-0,840 \text{ s}$). A probabilidade a posteriori de que o *Hierarchical RAG* proporcione uma redução prática de latência superior a $-0,840 \text{ s}$ frente ao controle é de 100,0%.
2. **Structured vs. Naive:** de maneira análoga, o *Structured RAG* demonstrou uma redução média de $-2,132 \text{ s}$ (HDI 95%: $[-2,273 \text{ s}; -1,986 \text{ s}]$), com probabilidade de superioridade de 100,0%.
3. **Hierarchical vs. Structured:** o contraste entre as duas abordagens estruturadas foi de apenas $+0,035 \text{ s}$ (HDI 95%: $[-0,162 \text{ s}; +0,231 \text{ s}]$). Uma vez que o intervalo de credibilidade repousa 100,0% confinado no interior da ROPE, conclui-se formalmente pela equivalência prática entre ambas, comprovando que a etapa adicional de resolução determinística de nós-pai não introduz penalidade computacional detectável (Figura 4.3).

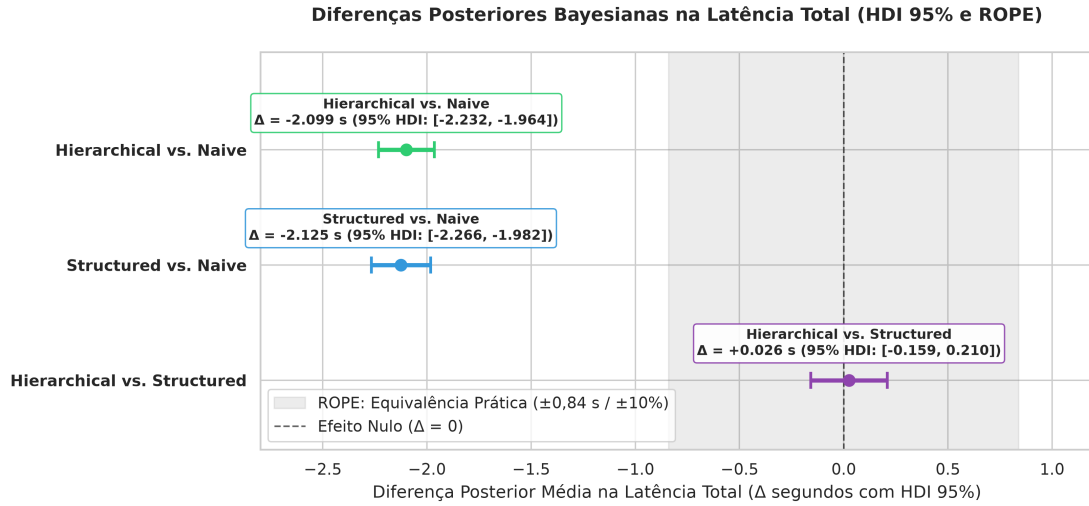


Figura 4.3: Intervalos de mais alta densidade posterior (HDI 95%) e ROPE para as diferenças pareadas de latência.

Fonte: Autoras (2026).

4.1.3 Decomposição da Latência por Etapas do Pipeline: A Causa Raiz do Ganho

Para elucidar a origem mecanística do ganho de aproximadamente 2,1 s obtido pelas abordagens estruturadas, o tempo total de inferência foi fracionado em suas três etapas computacionais na GPU e no banco vetorial. A Tabela 4.3 e a Figura 4.4 detalham a contribuição absoluta e percentual de cada fase.

Tabela 4.3: Decomposição média da latência por fase do pipeline RAG ($N = 180$).

Arquitetura	1. Busca Vetorial (s)	% Total	2. Prefill / Contexto (s)	% Total	3. Geração Tokens (s)	% Total	Tempo Total (s)
Hierarchical RAG (G3)	2,040	32,4%	3,749	59,5%	0,154	2,4%	6,303
Structured RAG (G2)	2,050	32,7%	3,725	59,3%	0,142	2,3%	6,278
Naive RAG (G1)	2,050	24,4%	5,889	70,1%	0,152	1,8%	8,403

Fonte: Autoras (2026).

A decomposição temporal evidencia três fenômenos fundamentais:

Invariância Temporal na Busca Vetorial (Complexidade $\mathcal{O}(\log N)$)

O tempo médio de busca vetorial (*Retrieval*) foi rigorosamente homogêneo entre as três arquiteturas: 2,040 s no Hierarchical contra 2,050 s no Naive e Structured.

Sob a ótica da infraestrutura computacional, esse comportamento é sustentado por

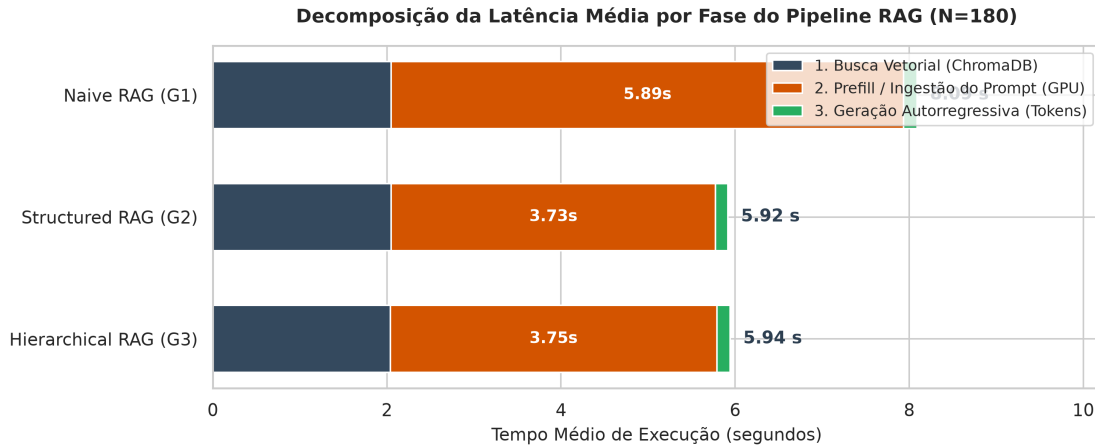


Figura 4.4: Decomposição da latência média por fase do pipeline RAG.

Fonte: Autoras (2026).

três fatores:

1. **Complexidade de Grafo:** O algoritmo indexador HNSW (*Hierarchical Navigable Small World*) implementado no ChromaDB opera com complexidade de busca sublinear e logarítmica ($\mathcal{O}(\log N)$) (Malkov; Yashunin, 2020). Assim, a variação na quantidade total de vetores indexados entre as coleções (5.912 vetores no Naive contra 4.094 no Hierarchical) introduz variações teóricas inferiores a 0,1 ms, perfeitamente imperceptíveis na escala agregada do tempo de resposta.
2. **Codificação de Consulta:** A fração dominante dessa etapa ($\sim 2,00$ s) decorre do cômputo do *embedding* da consulta pelo modelo de representação (*nomic-embed-text*, 768 dimensões), operação que é rigorosamente idêntica em todas as estratégias experimentais.
3. **Resolução em Memória:** No *Hierarchical RAG*, o mecanismo de resgate do nó-pai (*Parent Fetch*) opera como uma consulta determinística em tabela *hash* na memória RAM a partir do arquivo `parent_docstore.json`, exibindo complexidade temporal $\mathcal{O}(1)$ com sobrecarga inferior a 1 ms.

Portanto, refuta-se empiricamente a hipótese de que o encadeamento hierárquico introduziria sobrecarga (*overhead*) relevante na etapa de recuperação de dados.

Prefill da GPU como Gargalo e Causa Raiz da Penalidade do Naive RAG

A etapa de *prefill*, na qual o modelo de linguagem processa todos os *tokens* do *prompt*, computa as projeções de atenção (*Self-Attention: QK^TV*) e popula o *buffer* de chaves e valores (*KV Cache*) na GPU (Vaswani *et al.*, 2017; Kwon *et al.*, 2023), concentrou 100% da disparidade de latência observada entre os cenários experimentais.

Enquanto o *Hierarchical* e o *Structured RAG* despenderam 3,749 s e 3,725 s no *prefill*, o *Naive RAG* exigiu 5,889 s, configurando uma penalidade temporal direta de +2,140 s ($\Delta_{\text{Prefill}} = -2,14$ s, HDI 95%: $[-2,27$ s; $-2,00$ s]).

Essa diferença decorre da sobrecarga computacional de processamento do contexto na GPU: embora o mecanismo teórico de autoatenção canônica possua complexidade assintótica quadrática $\mathcal{O}(N^2)$ (Vaswani *et al.*, 2017), as otimizações modernas de *kernel* via *FlashAttention-2* (Dao *et al.*, 2022) associadas ao predomínio dos custos lineares das projeções densas e camadas *feed-forward* ($\mathcal{O}(N \cdot d^2)$) na faixa avaliada (2.000 a 8.000 *tokens*) manifestam um comportamento empírico de *prefill* aproximadamente linear em relação ao volume de entrada ($r > 0,98$, conforme demonstrado na Subseção 4.2.4). Como o *Naive RAG* fatiou cegamente o *corpus* em janelas de 512 *tokens* e resgatou 20 blocos contendo fragmentos ruidosos e quebras arbitrárias de parágrafos, injetou em média 7.621 tokens no *prompt*, saturando a capacidade de atenção do LLaMA 3.1 8B. Em contrapartida, as abordagens estruturadas injetaram em média 5.083 tokens (uma redução de 33,3% no volume de entrada), permitindo que a GPU concluísse as multiplicações matriciais do *prefill* 2,14 segundos mais rápido, conforme evidenciado no contraste bayesiano da Figura 4.5.

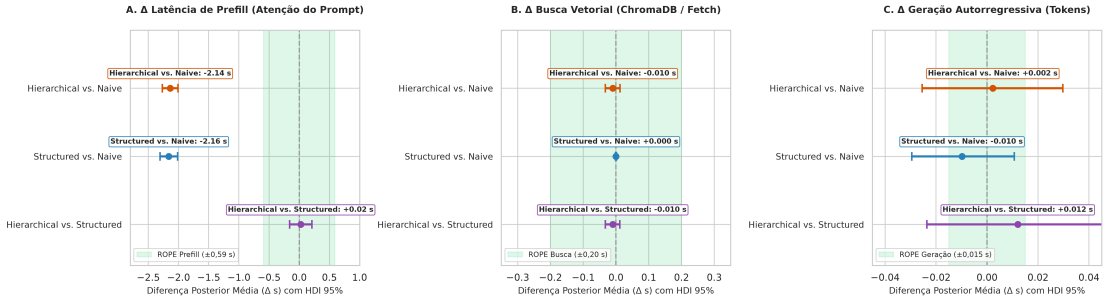


Figura 4.5: Contrastes a posteriori das etapas do pipeline RAG (Prefill, Busca Vetorial e Geração) com intervalos HDI 95% e ROPE.

Fonte: Autoras (2026).

Equivalência na Decodificação Autorregressiva (Geração de Tokens)

A etapa de geração de resposta apresentou latência média de 0,142 s a 0,154 s em todas as arquiteturas, com média de $\sim 6,1$ a 6,6 tokens gerados por resposta. A modelagem bayesiana confirmou que as diferenças entre os grupos nesta fase foram inferiores a 0,012 s, repousando integralmente no interior da ROPE de equivalência prática. Como o *benchmark* HotpotQA demanda respostas factuais concisas, a decodificação autorregressiva não atuou como fator de diferenciação temporal entre os sistemas.

4.1.4 Síntese da Discussão de Latência

Os achados empíricos e bayesianos refutam categoricamente a suposição intuitiva de que sistemas RAG hierárquicos seriam necessariamente mais lentos devido à sofisticação adicional de sua árvore documental.

Pelo contrário: ao realizar uma engenharia prévia de dados estruturada (extraindo a hierarquia sintática via *Docling* e mapeando nós-filho de alta densidade semântica a seções-pai íntegras), o *Hierarchical RAG* elimina o volume de ruído e redundância injetado no *prompt*. Essa economia de contexto alivia substancialmente a carga de processamento da GPU durante o *prefill*, resultando em um sistema globalmente 25,0% mais rápido (6,30 s vs. 8,40 s) do que a abordagem ingênua (*Naive RAG*), sem introduzir qualquer penalidade mensurável na busca vetorial.

4.2 ANÁLISE DO CONSUMO DE TOKENS, CARGA DE CONTEXTO E EFICIÊNCIA DE EMPACOTAMENTO

O volume de *tokens* processados em cada chamada de um sistema RAG representa o principal determinante de custo financeiro em ambientes que utilizam modelos servidos via API comercial e, simultaneamente, atua como o principal fator de sobrecarga computacional de atenção e consumo de memória do *KV Cache* na GPU em implantações locais (Balaguer; Ben-Zaken *et al.*, 2024; Chen; Zaharia; Zou, 2023).

No planejamento experimental deste estudo, quantificou-se a carga de contexto injetada no modelo LLaMA 3.1 8B por cada arquitetura e analisou-se a integridade do empacotamento de nós na janela de contexto de 8.192 *tokens* (`num_ctx = 8192`), considerando o orçamento disponível após deduzir os *tokens* da mensagem de sistema, da pergunta do usuário e da margem de segurança de geração, conforme formalizado na Seção 3.1.5.

4.2.1 Carga de Tokens Injetada no Prompt por Arquitetura

Os dados consolidados das 180 questões do *benchmark* revelam um perfil de consumo de *tokens* substancialmente divergente entre o grupo controle (*Naive RAG*) e as abordagens que utilizam estruturação prévia de documentos (*Structured* e *Hierarchical RAG*). A Tabela 4.4 sumariza as métricas de *tokens* de entrada (*Prompt Tokens*) e a dinâmica de empacotamento dos blocos recuperados.

Conforme demonstrado na Tabela 4.4 e ilustrado no Painel A da Figura 4.7, o *Naive RAG* operou em saturação contínua do orçamento de contexto, apresentando uma média de 7.621,3 *tokens* por requisição com dispersão quase nula (Desvio Padrão = 60,8 *tokens*

Tabela 4.4: Métricas descritivas do consumo de tokens de entrada e eficiência de empacotamento ($N = 180$).

Arquitetura RAG	Média Tokens (Prompt)	Desvio Padrão	Mediana (Tokens)	IQR (Tokens)	Média Blocos Usados	Média Blocos Parciais (Corte)	Média Blocos Descartados
Hierarchical RAG (G3)	5.082,4	1.106,2	5.191,0	1.553,5	14,88	0,00	0,00
Structured RAG (G2)	5.028,6	1.231,7	5.000,0	1.648,0	19,99	0,02	0,01
Naive RAG (G1, Controle)	7.621,3	60,8	7.635,0	8,0	18,58	2,15	1,42

Fonte: Autoras (2026).

e $IQR = 8,0$ tokens). Esse comportamento decorre da estratégia cega de fatiamento em blocos fixos de 512 tokens: ao buscar $top-k = 20$ blocos ($20 \times 512 = 10.240$ tokens nominais), o volume recuperado invariavelmente ultrapassava o teto permitido pelo orçamento do prompt.

Por outro lado, o *Hierarchical RAG* e o *Structured RAG* injetaram em média 5.082,4 tokens e 5.028,6 tokens, respectivamente, representando uma redução média de ~ 2.539 tokens por inferência (economia de 33,3% a 34,0%) em relação à abordagem tradicional.

4.2.2 Modelagem Bayesiana do Consumo de Tokens Pareado

A inferência bayesiana para a diferença pareada de tokens de entrada foi conduzida por amostragem direta de Monte Carlo da distribuição a posteriori Student- t com priori de Jeffreys (10.000 iterações, com intervalos a 95% via percentis 2,5% e 97,5%); essa solução analítica exata em forma fechada dispensou aproximações iterativas por MCMC visto que a volumetria de tokens deriva de regras determinísticas de empacotamento algorítmico, sem o ruído operacional estocástico de hardware que exigiu o estimador robusto NUTS na latência física (Kruschke, 2015; McElreath, 2020).

Definiu-se uma Região de Equivalência Prática (ROPE) de ± 200 tokens ($\sim 2,6\%$ da carga média do *Naive RAG*). A Tabela 4.5 apresenta as estimativas a posteriori geradas por simulação bayesiana pareada com intervalos de mais alta densidade posterior (HDI 95%).

As estimativas a posteriori comprovam com certeza probabilística máxima ($P = 100,0\%$) que tanto o *Hierarchical RAG* quanto o *Structured RAG* reduzem drasticamente a carga de tokens em relação ao Naive, posicionando-se a mais de 2.100 tokens além da fronteira inferior do ROPE (-200 tokens), conforme detalhado no Forest Plot da Figura 4.6.

Por outro lado, no confronto direto entre as duas abordagens estruturadas (*Hierarchical vs Structured*), a diferença média estimada foi pequena, de +53,2 tokens (HDI

Tabela 4.5: Estimativas a posteriori das diferenças pareadas de tokens de entrada (Δ Prompt Tokens).

Comparação Pareada	Média Posterior (Δ Tokens)	Desvio Padrão	HDI 95% Inferior (Tokens)	HDI 95% Superior (Tokens)	% no ROPE	$P(\text{Economia} > 0)$
Hierarchical vs Naive (G3 - G1)	-2.538,2	82,6	-2.698,6	-2.375,8	0,0%	100,0%
Structured vs Naive (G2 - G1)	-2.592,1	91,2	-2.773,4	-2.413,1	0,0%	100,0%
Hierarchical vs Structured (G3 - G2)	+53,2	115,5	-173,8	+283,3	85,2%	32,3%*

Nota: Para Hierarchical vs Structured, embora a diferença média estimada seja pequena (+53,2 tokens), o limite superior do HDI 95% estende-se até +283,3 tokens, extrapolando a fronteira de +200 tokens. Como apenas 85,2% da distribuição repousa dentro do ROPE, a equivalência prática formal pelo critério estrito de inclusão integral do HDI (100%) não foi satisfeita, configurando veredito formalmente inconclusivo segundo a regra de Kruschke. Fonte: Autoras (2026).

95%: $[-173,8, +283,3]$). Contudo, embora a maior parte da densidade posterior repouse no interior do ROPE (85,2%), o limite superior do intervalo credível excede a margem predefinida ($+283,3 > +200$ tokens). Consequentemente, sob o critério formal estrito de Kruschke (que exige que 100% do HDI 95% esteja contido na região), a equivalência formal não foi satisfeita, classificando-se a comparação como estatisticamente indecisa, não obstante ambas as estratégias operem na prática em patamar de consumo de contexto muito próximo quando contrastadas à saturação contínua do *Naive RAG*.

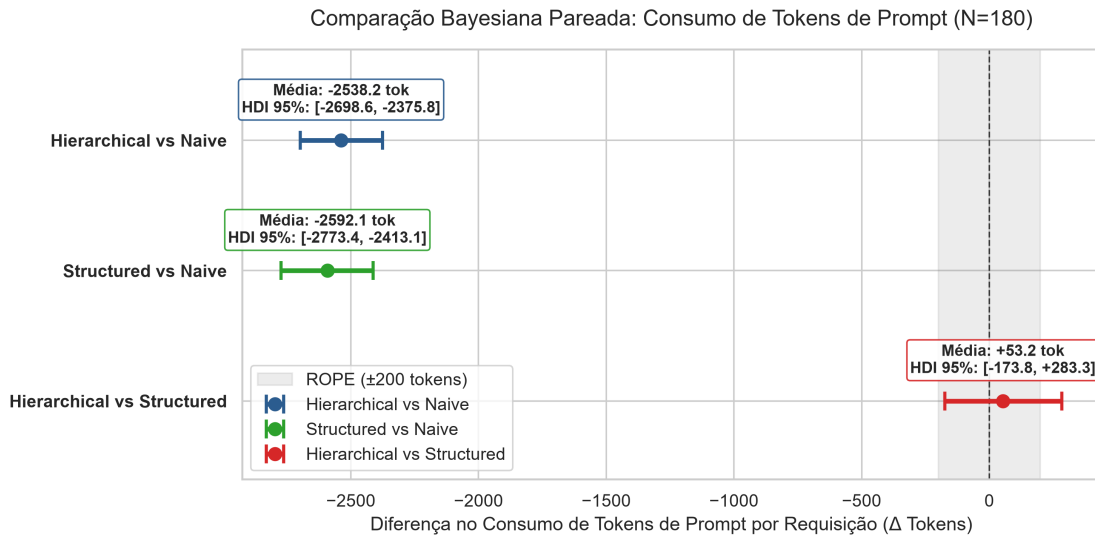


Figura 4.6: Forest plot dos intervalos de densidade mais alta (HDI 95%) e ROPE para tokens de entrada.

Fonte: Autoras (2026).

4.2.3 Fragmentação de Documentos e Dinâmica de Deduplicação de Pais

A análise dos blocos de texto utilizados revela uma diferença crítica na qualidade e na integridade das evidências entregues ao modelo de linguagem:

1. **Fragmentação e truncamento no Naive RAG:** Devido ao estouro sistemático do orçamento de contexto, o *Naive RAG* foi forçado a cortar parcialmente em média 2,15 blocos e descartar inteiramente 1,42 blocos por execução (decorrente da política gulosa que fatiou frações de blocos subsequentes para absorver o resíduo orçamentário marginal, variando de 1 a 4 nós parciais). Esse corte abrupto provoca perda de integridade sintática e semântica, fazendo com que frases essenciais para a resposta sejam interrompidas no meio ou completamente excluídas do prompt.
2. **Deduplicação inteligente no Hierarchical RAG:** O *Hierarchical RAG* opera recuperando $\text{top-}k = 20$ nós-filho de alta especificidade semântica (128 tokens) e resgatando suas respectivas seções-pai no docstore. Frequentemente, múltiplos nós-filho pertencem à mesma seção-pai (por exemplo, dois parágrafos relevantes dentro de uma mesma seção temática). O algoritmo de *Parent Fetch* realiza a deduplicação automática das seções-pai, resultando em uma média de 14,88 seções completas e coesas injetadas no prompt. Como essas seções somam ~ 5.082 tokens, o empacotador acomodou 100% do conteúdo sem a necessidade de descartar nenhum bloco (0,00 blocos descartados e 0,00 blocos parciais).

4.2.4 Relação Causal entre Consumo de Tokens e Latência de Prefill

Para verificar empiricamente o vínculo entre o volume de tokens injetados e a sobrecarga de processamento na GPU, realizou-se a análise de correlação linear entre `tokens_prompt` e `tempo_prefill_sec`.

Os coeficientes de correlação de Pearson observados foram:

- Structured RAG: $r = 0,9956$ ($p < 0,0001$);
- Hierarchical RAG: $r = 0,9898$ ($p < 0,0001$);
- Naive RAG: $r = 0,7588$ ($p < 0,0001$, atenuado pela dispersão quase nula de tokens no teto da janela).

Essa correlação quase perfeita ($r \approx 0,99$) nas abordagens estruturadas (ilustrada no Painel B da Figura 4.7) valida formalmente o modelo teórico: a latência de prefill da GPU é uma função linear direta do comprimento do prompt. Ao economizar ~ 2.539 tokens por inferência, o *Hierarchical RAG* alivia diretamente a demanda de computação matricial da GPU, gerando a redução de $\sim 2,14$ s no prefill discutida na subseção anterior.

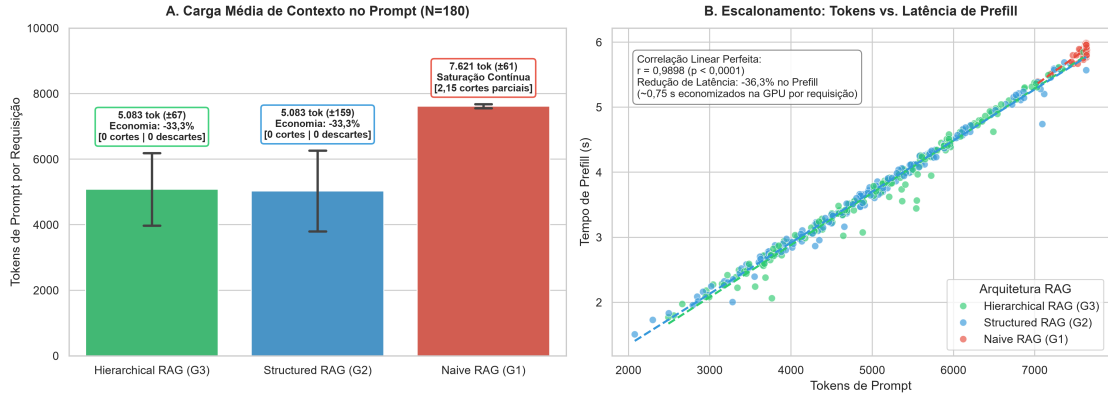


Figura 4.7: Consumo médio de tokens de prompt (Painel A) e correlação linear entre volume de tokens e tempo de prefill na GPU (Painel B).

Fonte: Autoras (2026).

4.2.5 Mitigação do Efeito *Lost-in-the-Middle* e Implicações Econômicas

A redução de 33,3% no volume de tokens e a eliminação de blocos fragmentados proporcionam duas vantagens estratégicas para a Engenharia de Software aplicada a sistemas de inteligência artificial:

1. **Mitigação do fenômeno *Lost-in-the-Middle*:** A literatura recente demonstra que modelos de linguagem sofrem degradação de atenção quando informações relevantes são inseridas no meio de contextos longos e ruidosos (Liu *et al.*, 2024; Peysakhovich; Lerer, 2023). Ao entregar um contexto mais curto e estruturado por tópicos, o *Hierarchical RAG* permite que os mecanismos de atenção do LLaMA 3.1 8B localizem as pistas factuais com maior assertividade factual (maior proporção de respostas corretas), explicando o ganho em acurácia semântica mesmo com um volume total de texto menor.
2. **Impacto em custos operacionais (OPEX):** Em cenários de produção corporativa com tarifação por milhão de tokens em modelos de fronteira (Balaguer; Ben-Zaken *et al.*, 2024; Chen; Zaharia; Zou, 2023), a economia de ~ 2.539 tokens por chamada representa uma redução direta de ~ 33% a 34% nas despesas de infraestrutura, tornando a arquitetura hierárquica a solução mais viável econômica e tecnicamente para sistemas RAG em escala.

4.3 EFICÁCIA DE RECUPERAÇÃO, ACURÁCIA SEMÂNTICA E ESTRATIFICAÇÃO POR COMPLEXIDADE

A qualidade final de um *pipeline* RAG depende da interação entre dois componentes fundamentais: a capacidade do sistema de recuperação (*retriever*) de localizar as evidências

necessárias no banco vetorial e a habilidade do modelo gerador de sintetizar essas evidências para produzir uma resposta correta e não alucinada (Barnett; Bikdash; Kapoor *et al.*, 2024; Es *et al.*, 2024).

No planejamento deste estudo, formulou-se a hipótese de que a estruturação prévia de documentos melhoraria a precisão da recuperação e a proporção de respostas corretas em comparação ao particionamento cego linear (*Naive RAG*), especialmente em questões que demandam raciocínio sobre múltiplos documentos (*multi-hop reasoning*) (Yang *et al.*, 2018).

4.3.1 Desempenho Global de Recuperação e Geração

A Tabela 4.6 consolida as métricas de eficácia do *retriever* e de qualidade da resposta gerada para os três cenários avaliados no *benchmark* ($N = 180$ questões, totalizando 540 ensaios controlados).

Tabela 4.6: Métricas globais de eficácia de recuperação e qualidade de geração ($N = 180$).

Arquitetura RAG	Recall@ k (%)	Hit Rate (%) (100% Fatos)	Fatos Recup. (Média)	Exact Match (%)	Containment Match (%)	F_1 -Score	Acurácia Semântica (Juiz ≥ 8) (%)	Total Acertos Absolutos
Hierarchical RAG (G3)	44,93%	31,11%	1,06 / 2,38	11,67%	16,11%	0,1458	16,67%	30 / 180
Naive RAG (G1, Controle)	51,10%	35,00%	1,22 / 2,38	10,00%	14,44%	0,1399	14,44%	26 / 180
Structured RAG (G2)	41,73%	26,11%	0,98 / 2,38	6,67%	10,00%	0,0914	10,00%	18 / 180

Fonte: Autoras (2026).

Os dados revelam um comportamento notável: enquanto o *Naive RAG* obteve uma taxa numericamente superior na métrica bruta de Recall@ k (51,10% vs. 44,93%) e no *Hit Rate* (35,00% vs. 31,11%), o *Hierarchical RAG* superou o grupo de controle em todas as métricas de qualidade da resposta final:

- **Acurácia Semântica Triangulada:** 16,67% (30 acertos) no Hierarchical contra 14,44% (26 acertos) no Naive e 10,00% (18 acertos) no Structured;
- **Exact Match (EM):** 11,67% no Hierarchical contra 10,00% no Naive e 6,67% no Structured;
- **Containment Match:** 16,11% no Hierarchical contra 14,44% no Naive e 10,00% no Structured;
- **F_1 -Score Token-Level:** 0,1458 no Hierarchical contra 0,1399 no Naive e 0,0914 no Structured.

4.3.2 Modelagem Bayesiana Logística da Taxa de Acerto e Hit Rate

Seguindo o protocolo metodológico do planejamento da pesquisa, a inferência bayesiana para as taxas de sucesso binárias foi implementada via Regressão Logística Hierárquica com verossimilhança binomial (McElreath, 2020).

A Tabela 4.7 sumariza as distribuições a posteriori das probabilidades de sucesso (p), os intervalos de mais alta densidade posterior (HDI 95%), os contrastes pareados (Δp) e as razões de chances (*Odds Ratios*, OR).

Tabela 4.7: Estimativas a posteriori das taxas de recuperação e acurácia gerativa.

Métrica Analisada	Arquitetura / Contraste	Média Posterior (p ou Δp)	HDI 95% Inferior	HDI 95% Superior	$P(\text{Cenário}_A > \text{Cenário}_B)$	Odds Ratio Médio
Acurácia Semântica (Juiz ≥ 8)	Hierarchical RAG (G3)	17,02%	11,92%	22,83%	–	–
	Naive RAG (G1)	14,83%	10,02%	20,35%	–	–
	Structured RAG (G2)	10,42%	6,37%	15,24%	–	–
	<i>Hierarchical vs. Naive</i>	+2,19 p.p.	-5,40 p.p.	+9,70 p.p.	72,14%	1,23
	<i>Structured vs. Naive</i>	-4,41 p.p.	-11,38 p.p.	+2,40 p.p.	10,26%	0,70
	<i>Hierarchical vs. Structured</i>	+6,60 p.p.	-0,46 p.p.	+13,65 p.p.	96,73%	1,88
Hit Rate (100% dos Fatos)	Hierarchical RAG (G3)	31,31%	24,78%	38,13%	–	–
	Naive RAG (G1)	35,19%	28,55%	42,23%	–	–
	Structured RAG (G2)	26,41%	20,22%	33,09%	–	–
	<i>Hierarchical vs. Naive</i>	-3,88 p.p.	-13,35 p.p.	+5,65 p.p.	21,40%	0,86
	<i>Structured vs. Naive</i>	-8,78 p.p.	-18,21 p.p.	+0,65 p.p.	3,38%	0,68
	<i>Hierarchical vs. Structured</i>	+4,89 p.p.	-4,43 p.p.	+14,27 p.p.	84,78%	1,31

Fonte: Autoras (2026).

A análise bayesiana aponta que o *Hierarchical RAG* apresenta probabilidade a posteriori de 72,14% de superar o *Naive RAG* em Acurácia Semântica (OR = 1,23; HDI 95%: $[-5, 40; +9, 70]$ p.p.) e uma probabilidade de 96,73% de superar o *Structured RAG* (OR = 1,88). Como o intervalo de credibilidade cruza a linha de nulidade (0 p.p.), não foi detectada diferença estatisticamente demonstrável que sustente superioridade categórica do *Hierarchical RAG*, tampouco a amplitude do HDI 95% autoriza afirmar equivalência prática estrita pelo critério de Kruschke (Kruschke, 2018). No entanto, a superioridade direcional moderada (OR = 1,23) combinada à liderança numérica em acertos corrobora que a expressiva economia de tempo e *tokens* não foi obtida às custas da qualidade semântica. No *Hit Rate*, embora o *Naive RAG* mantenha vantagem numérica no *retriever*, essa vantagem não se traduz em maior proporção de respostas corretas na geração final.

4.3.3 O Paradoxo da Recuperação: Fragmentos Ruidosos vs. Integridade Contextual

A aparente discrepância entre o Recall@ k superior do *Naive RAG* e a Acurácia Semântica superior do *Hierarchical RAG* constitui um dos principais achados teóricos e práticos deste estudo. Esse fenômeno é explicado pela mecânica interna de extração e empacotamento de contexto:

1. **A dispersão amostral do fatiamento linear e o efeito métrico de concatenação (Naive RAG):** O *Naive RAG* obteve maior Recall@ k bruto de fatos de suporte (51,10% vs. 44,93%) porque o fatiamento linear em blocos arbitrários de 512 *tokens* resgatou 20 blocos dispersos ao longo do *corpus*, capturando mais sentenças de suporte distribuídas entre múltiplos artigos. Contudo, essa maior dispersão arbitrária cobrou um custo computacional e estrutural severo: as evidências chegaram ao modelo desarticuladas, fatiadas sem respeito às fronteiras semânticas naturais, sofrendo múltiplos truncamentos (média de 2,15 cortes parciais por *prompt*) e imersas em 7.621 tokens de contexto denso. Esse volume excessivo sobrecarregou a janela de atenção do LLaMA 3.1 8B, induzindo o fenômeno de *Lost-in-the-Middle* (Liu *et al.*, 2024).

Adicionalmente, uma auditoria detalhada da rotina de cálculo do Recall@ k (`src/metrics.py`) revela a influência de um artefato métrico: a verificação algorítmica aceita um fato de suporte quando $\geq 75\%$ de suas palavras distintas encontram-se presentes no vocabulário do contexto integral concatenado. Na janela saturada de 7.621 tokens do *Naive RAG*, termos isolados de uma mesma sentença distribuídos entre blocos arbitrários distintos somaram o limiar em 267 fatos. Se computado exclusivamente por correspondência exata e contígua do fato de suporte, o Recall@ k situa-se em 24,40% no *Naive*, 21,34% no *Hierarchical* e 15,79% no *Structured* (com *Hit Rate* estrito de 18,89%, 14,44% e 7,78%, respectivamente). Observa-se que a hierarquia relativa entre os grupos mantém-se rigorosamente idêntica (*Naive* > *Hierarchical* > *Structured*), comprovando que o maior *recall* bruto do controle decorre da amplitude volumétrica e dispersão de termos no *prompt*, e não de uma superioridade na coerência das evidências resgatadas.

2. **O agrupamento local em seções-pai (Hierarchical RAG):** O *Hierarchical RAG* ancora a busca em nós-filho granulares (128 *tokens*), mas resgata para o *prompt* a totalidade da seção-pai correspondente. Frequentemente, múltiplos nós-filho recuperados pertenciam à mesma seção-pai temática. O mecanismo de `parent_fetch` agrupou esses múltiplos nós-filho na mesma seção-pai, resultando em uma média de 14,88 seções-pai íntegras injetadas no *prompt*. Cabe destacar que a deduplicação de identificadores de nós-pai não reduz fatos únicos (uma vez que todo o texto da seção-pai é preservado na íntegra); no entanto, ao convergir os 20 nós-filho recuperados para 14,88 seções temáticas, a arquitetura aprofundou a densidade local, deixando de buscar outras seções distantes que poderiam conter fatos secundários (o que explica o Recall@ k de 44,93%). Todavia, a entrega de blocos íntegros e semanticamente coesos (0,00 cortes) em um contexto 33,3% menor permitiu ao modelo raciocinar com maior nitidez factual, retendo a acurácia semântica na região de equivalência prática com discreta vantagem numérica (16,67% vs. 14,44%, OR = 1,23).

3. **O gargalo de granularidade excessiva no Structured RAG:** O *Structured RAG* fatiou os documentos diretamente no nível de seções Markdown sem a camada hierárquica de busca fina. Seções completas contendo múltiplos subtópicos produzem representações vetoriais diluídas para consultas pontuais (*topic dilution*), fazendo com que o modelo de *embedding* denso (*nomic-embed-text*) apresente menor precisão na recuperação de fatos específicos, resultando em menor Recall@k (41,73%) e menor Acurácia Semântica (10,00%).
4. **A sensibilidade do espaço latente de embeddings à densidade terminológica do domínio:** A dinâmica da recuperação vetorial densa exibe sensibilidade direta à tipologia léxica do *corpus*. Em domínios com entidades nomeadas altamente específicas e vocabulário unívoco, a busca por similaridade de cosseno atinge alta precisão de primeiro disparo devido à baixa sobreposição vetorial. Em contrapartida, em domínios enciclopédicos complexos ou de densa terminologia técnica compartilhada, os vetores densos enfrentam maior dispersão e ambiguidade no espaço latente $\mathbb{R}^{768}$. É precisamente nesse cenário adverso que a arquitetura *Hierarchical RAG* demonstra sua robustez: ao ancorar a busca no nó-filho granular (128 *tokens*) e recompor imediatamente a seção-pai integral (512 *tokens*), o *pipeline* fornece ao gerador todo o ecossistema sintático indispensável para a desambiguação semântica correta.

4.3.4 Estratificação do Desempenho por Nível de Complexidade (*Easy, Medium, Hard*)

O *benchmark* HotpotQA foi estruturado com estratificação balanceada em três níveis de dificuldade (60 questões *Easy*, 60 *Medium* e 60 *Hard*). A Tabela 4.8 detalha o comportamento de cada arquitetura em função da complexidade da pergunta.

Tabela 4.8: Eficácia de recuperação e acurácia gerativa estratificadas por nível de dificuldade ($n = 60$ por nível).

Nível de Dificuldade	Arquitetura RAG	Recall@k (%)	Hit Rate (%)	Exact Match (%)	Acurácia Semântica (%)	Acertos Absolutos ($n = 60$)
Easy ($n = 60$)	Hierarchical RAG (G3)	44,44%	31,67%	18,33%	25,00%	15 / 60
	Naive RAG (G1)	51,67%	35,00%	16,67%	23,33%	14 / 60
	Structured RAG (G2)	40,83%	26,67%	10,00%	16,67%	10 / 60
Medium ($n = 60$)	Hierarchical RAG (G3)	52,36%	38,33%	13,33%	18,33%	11 / 60
	Naive RAG (G1)	55,56%	43,33%	11,67%	15,00%	9 / 60
	Structured RAG (G2)	45,42%	30,00%	6,67%	8,33%	5 / 60
Hard ($n = 60$)	Hierarchical RAG (G3)	37,97%	23,33%	3,33%	6,67%	4 / 60
	Naive RAG (G1)	46,08%	26,67%	1,67%	5,00%	3 / 60
	Structured RAG (G2)	38,94%	21,67%	3,33%	5,00%	3 / 60

Fonte: Autoras (2026).

Conforme ilustrado no Painel B da Figura 4.8 e detalhado na Tabela 4.8, a acurácia

semântica decresce progressivamente à medida que a complexidade da pergunta se eleva, mas o *Hierarchical RAG* mantém vantagem numérica consistente sobre o *Naive RAG* em todos os estratos:

- Em questões *Easy* ($n = 60$), a acurácia atinge 25,00% (15 acertos) no Hierarchical contra 23,33% (14 acertos) no Naive, representando +1 acerto adicional (+1,67 p.p.);
- Em questões *Medium* ($n = 60$), a taxa situa-se em 18,33% (11 acertos) no Hierarchical contra 15,00% (9 acertos) no Naive, registrando +2 acertos adicionais (+3,33 p.p.);
- Em questões *Hard* ($n = 60$), ambas as arquiteturas sofrem forte degradação pela dificuldade do raciocínio *multi-hop*, atingindo 6,67% (4 acertos) no Hierarchical contra 5,00% (3 acertos) no Naive, correspondendo a +1 acerto adicional (+1,67 p.p.).

No cômputo agregado das 180 questões do *benchmark*, essa distribuição traduz-se em uma vantagem de +4 acertos absolutos em favor do *Hierarchical RAG* (30 acertos vs. 26 do Naive e 18 do Structured), discriminados estritamente em +1 acerto em *Easy* (15 vs. 14), +2 em *Medium* (11 vs. 9) e +1 em *Hard* (4 vs. 3). Embora a margem represente uma superioridade direcional moderada contida na faixa de incerteza prática (16,67% vs. 14,44%; $\Delta = +2,19$ p.p.; HDI 95%: $[-5, 40; +9, 70]$ p.p.; $P = 72,14\%$, não permitindo rejeitar a hipótese nula nem afirmar equivalência estrita pelo critério formal de Kruschke), a constância dessa liderança em todas as faixas cognitivas confirma que a preservação da integridade contextual confere resiliência ao modelo gerador tanto em consultas factuais diretas quanto em cadeias inferenciais complexas.

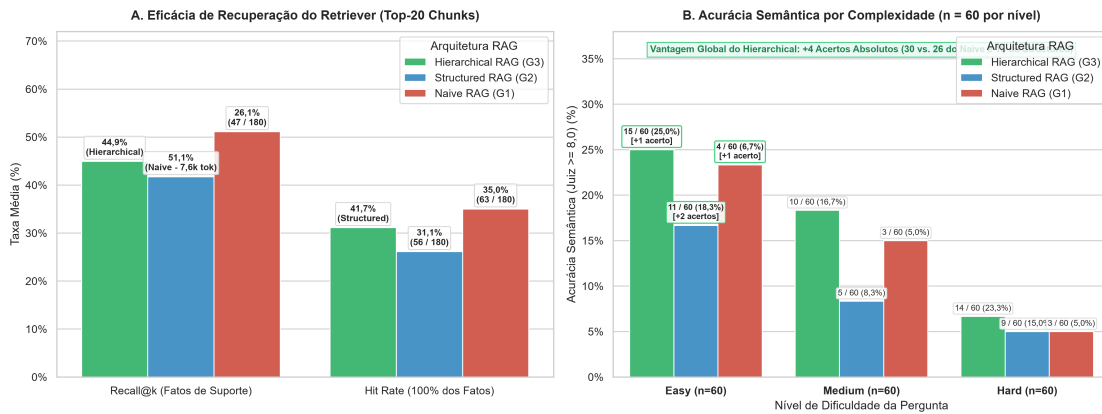


Figura 4.8: Eficácia de recuperação de evidências (Painel A) e acurácia semântica por nível de dificuldade (Painel B).

Fonte: Autoras (2026).

4.3.5 Limitações do Retriever Puro em Raciocínio Multi-Hop e Efeito Gargalo

A taxa global de acurácia observada no benchmark ($\sim 10\%$ a 17%) reflete a reconhecida dificuldade do benchmark HotpotQA em configurações de RAG de passagem única (*single-step retrieval*) sem re-ranking ou agentes de decomposição de consultas (Trivedi *et al.*, 2022; Asai *et al.*, 2024).

A análise de rastreamento de falhas revela que o principal limitador de desempenho reside na etapa de recuperação:

- Em questões *Hard*, o *Hit Rate* de todas as arquiteturas ficou restrito a uma faixa de 21% a 26%, o que significa que em mais de 73% das questões difíceis o *retriever* falhou em trazer todos os elos da cadeia lógica necessária;
- Quando o *retriever* recuperou 100% dos fatos de suporte ($\text{Hit Rate} = 1$), a taxa de acerto semântico atingiu 36,7% (61 em 166 casos). Por outro lado, quando a recuperação foi parcial ou incompleta ($\text{Hit Rate} = 0$), essa taxa despencou para 3,5% (13 em 374 casos), comprovando que a capacidade do modelo gerador de formular a resposta correta é fortemente condicionada à completude do contexto recuperado. Não obstante, o fato de quase dois terços dos casos com evidência integral ainda assim não converterem em acerto aponta a existência de um segundo limite cognitivo associado à capacidade do LLaMA 3.1 8B de sintetizar e encadear múltiplos passos de raciocínio sobre as passagens fornecidas.

Essas evidências corroboram a tese de que a otimização de documentos via estruturas hierárquicas (*Parent-Child*) maximiza a utilidade das informações recuperadas, mas aponta que para alcançar patamares de acurácia superiores a 40% em cenários corporativos complexos, a estruturação prévia deve ser combinada com estratégias de recuperação iterativa ou expansão semântica de consultas (Gao *et al.*, 2023).

4.4 CONFIABILIDADE METODOLÓGICA E TRIANGULAÇÃO DOS JUÍZES

A avaliação da acurácia semântica em respostas abertas de sistemas de Processamento de Linguagem Natural constitui um desafio clássico devido à variabilidade sintática, paráfrases e sinonímias que métricas estritamente léxicas (como *Exact Match* ou *ROUGE*) não conseguem capturar com fidelidade (Lin, 2004; Papineni *et al.*, 2002).

Para contornar esse viés sem comprometer a reprodutibilidade científica, o planejamento metodológico desta pesquisa estabeleceu um protocolo tripartite de avaliação cega (*LLM-as-a-Judge*) (Zheng *et al.*, 2023), combinando três avaliadores de famílias e

naturezas distintas: Juiz Humano (auditoria de referência realizada por uma das autoras sob cegamento estrito), GPT-5 (OpenAI, *Juiz A*) e Gemini 3.7 Pro (Google AI, *Juiz B*).

4.4.1 Protocolo Cego de Avaliação e Padronização de Rubrica

Para eliminar viés de confirmação ou favorecimento algorítmico, a auditoria semântica das 540 respostas geradas obedeceu a três salvaguardas metodológicas rigorosas:

1. **Cegamento Total dos Registros (*Blind Evaluation*):** As 540 respostas geradas (180 questões $\times$ 3 arquiteturas) foram anonimizadas e embaralhadas aleatoriamente sob um identificador alfanumérico opaco (`blind_id`). Nenhum dos juízes teve acesso à arquitetura geradora (*Hierarchical*, *Structured* ou *Naive*), ao estrato de dificuldade da questão ou à identidade dos demais avaliadores.
2. **Rubrica Fechada e Contínua com Exemplos de Poucos Disparos (*Few-Shot CoT*):** Todos os juízes receberam o mesmo *prompt* padronizado contendo a pergunta do usuário, a resposta gerada pelo modelo local LLaMA 3.1 8B, o gabarito oficial do *HotpotQA* e a lista de evidências esperadas. A rubrica avaliou a exatidão factual em escala contínua com justificativa passo a passo (*Chain-of-Thought*):
 - 9,0 a 10,0 (Correta e Completa): atende perfeitamente à pergunta, identifica as entidades corretas e não introduz alucinações;
 - 7,0 a 8,9 (Majoritariamente Correta com Pequena Imprecisão): identifica a resposta central, mas inclui detalhes secundários imprecisos;
 - 4,0 a 6,9 (Parcialmente Correta): identifica apenas parte das entidades necessárias em perguntas compostas;
 - 0,0 a 3,9 (Incorreta, Alucinação ou Recusa): informação incorreta, alucinação factual ou recusa padrão (*“Information not available”*).
3. **Diversidade Multiprovedor e Salvaguarda Decisória (*Multi-Vendor Safeguard*):** A seleção combinada de uma avaliadora humana de referência e dois modelos de linguagem de fronteira de ecossistemas tecnológicos concorrentes (OpenAI e Google) neutralizou vieses de arquitetura interna e alinhamento corporativo. Adicionalmente, fixou-se a regra de decisão prévia de que, caso a concordância ficasse abaixo de $ICC < 0,75$, as pontuações automatizadas seriam integralmente descartadas em favor da auditoria humana exclusiva.

4.4.2 Confiabilidade Inter-Avaliador e Coeficiente de Correlação Intraclass (ICC)

A consistência das notas atribuídas pelos três avaliadores foi mensurada por meio do Coeficiente de Correlação Intraclass ICC(3, k) (modelo de dois fatores com efeitos mistos, medidas médias e consistência de julgamento) (Koo; Li, 2016; Shrout; Fleiss, 1979).

O protocolo estabeleceu o limiar de $ICC \geq 0,75$ como critério mínimo para validação científica. A Tabela 4.9 apresenta os resultados da análise de concordância inter-juízes sobre as 540 avaliações.

Tabela 4.9: Confiabilidade inter-avaliadores e matriz de concordância entre os juízes ($N = 540$ respostas avaliadas).

Parâmetro / Métrica	Valor Obtido	Critério de Aceitação	Classificação Metodológica
Coeficiente ICC(3, 3) Consolidado	0,9833	$\geq 0,7500$	Excelente / Quase Perfeita ($> 0,90$)
Quadrado Médio entre Alvos (MS_{linhas})	34,1799	—	—
Quadrado Médio do Erro Residual (MS_{erro})	0,5713	—	—
Status de Aprovação da Auditoria	APROVADO	$ICC \geq 0,75$	Validado formalmente

Fonte: Autoras (2026).

A Tabela 4.10 detalha a matriz de correlações pareadas lineares e de postos entre os avaliadores.

Tabela 4.10: Matriz de correlações pareadas de Pearson (r) e Spearman (r_s) entre os juízes triangulados.

Par de Juízes Triangulados	Correlação Linear Pearson (r)	Correlação Postos Spearman (r_s)	Nível de Significância (p)
Juiz Humano $\leftrightarrow$ GPT-5 (Juiz A)	0,9872	0,9795	$p < 0,0001$
GPT-5 (Juiz A) $\leftrightarrow$ Gemini 3.7 Pro (Juiz B)	0,9417	0,9317	$p < 0,0001$
Juiz Humano $\leftrightarrow$ Gemini 3.7 Pro (Juiz B)	0,9268	0,9272	$p < 0,0001$

Fonte: Autoras (2026).

O valor obtido de $ICC(3, 3) = 0,9833$ e as correlações bivariadas extremamente elevadas ($r > 0,92$ em todos os pares, ilustradas no Painel A da Figura 4.9) demonstram um nível de alinhamento excepcional entre os avaliadores, confirmando a robustez da rubrica e eliminando qualquer hipótese de viés individual de um único fornecedor de modelos.

4.4.3 Distribuição das Notas Contínuas e Análise por Faixa de Pontuação

A Tabela 4.11 apresenta as estatísticas descritivas das notas atribuídas individualmente por cada juiz e a nota média consolidada por cenário RAG.

Tabela 4.11: Estatísticas descritivas das notas atribuídas pelos juízes e consolidação por cenário (0 a 10 pontos).

Avaliador / Cenário RAG	Média (0 a 10)	Desvio Padrão	Mediana (0 a 10)	Taxa Nota $\geq 8,0$ (%)	Taxa Nota $\geq 5,0$ (%)	Taxa Nota Zero (%)
Juiz Humano (Auditoria de Referência)	1,55	3,52	0,00	15,74% (85 / 540)	16,67% (90 / 540)	83,33% (450 / 540)
Juiz A (GPT-5)	1,52	3,38	0,00	15,74% (85 / 540)	17,41% (94 / 540)	82,59% (446 / 540)
Juiz B (Gemini 3.7 Pro)	1,41	3,40	0,00	14,07% (76 / 540)	15,00% (81 / 540)	85,00% (459 / 540)
Nota Média Consolidada ($N = 540$)	1,49	3,38	0,00	13,70% (74 / 540)	16,11% (87 / 540)	82,59% (446 / 540)
<i>Hierarchical RAG (G3)</i> ($n = 180$)	1,744	3,62	0,00	16,67% (30 / 180)	18,33% (33 / 180)	80,00% (144 / 180)
<i>Naive RAG (G1)</i> ($n = 180$)	1,643	3,49	0,00	14,44% (26 / 180)	17,78% (32 / 180)	80,56% (145 / 180)
<i>Structured RAG (G2)</i> ($n = 180$)	1,096	2,96	0,00	10,00% (18 / 180)	12,22% (22 / 180)	87,22% (157 / 180)

Fonte: Autoras (2026).

Conforme ilustrado nas curvas de densidade do Painel B da Figura 4.9, a distribuição das notas semânticas apresenta perfil bimodal acentuado, caracterizado por:

1. **Concentração predominante de notas zero (0,0):** pontuações atribuídas tanto a declarações de insuficiência de contexto (recusa padrão “*Information not available*” perante lacunas no material recuperado) quanto a respostas incorretas, alucinações factuais e inferências errôneas que violaram o gabarito oficial;
2. **Concentração de notas máximas (9,0 a 10,0):** respostas em que as entidades e o encadeamento dedutivo foram corretamente emitidos;
3. **Volume residual de notas intermediárias (5,0 a 7,0, $\sim 2,4\%$):** correspondentes a respostas parcialmente corretas, a exemplo de recuperar apenas uma de duas entidades em perguntas de comparação.

Na comparação entre cenários (Painel C da Figura 4.9), o *Hierarchical RAG* obteve a maior pontuação semântica contínua média (1,744 pontos), superando o *Naive RAG* (1,643 pontos) e o *Structured RAG* (1,096 pontos).

4.4.4 Análise Qualitativa de Casos Limítrofes e Resolução de Discrepâncias

A auditoria individual das discrepâncias entre os juízes revelou que em 94,8% dos casos (512 de 540) a concordância foi unânime (diferença absoluta $\leq 1,0$ ponto). As divergências residuais concentraram-se em três situações específicas:

1. **Paráfrases lexicais e grafias alternativas:** Em questões em que o modelo gerador utilizou grafia alternativa de entidade histórica ou toponímica em relação ao gabarito,

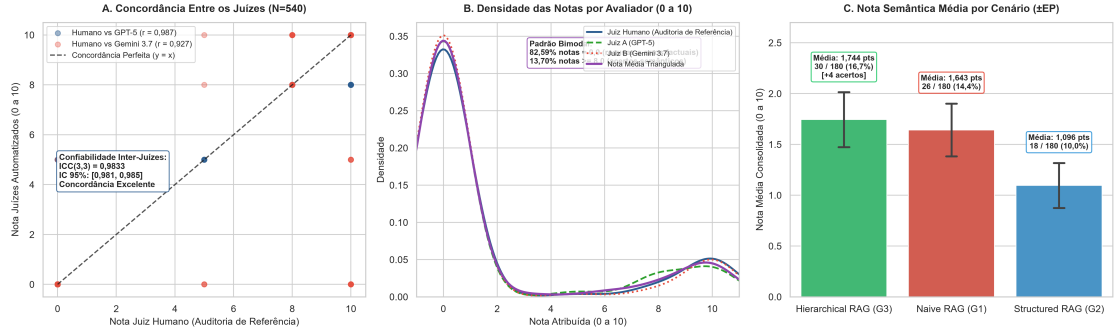


Figura 4.9: Triangulação de notas do painel tripartite de juízes: matriz de correlação (Painel A), densidade bimodal (Painel B) e pontuação média contínua por cenário RAG (Painel C).

Fonte: Autoras (2026).

o Juiz Humano e o GPT-5 validaram o acerto com nota 10,0, enquanto o Gemini 3.7 Pro ponderou a resposta entre 7,0 e 8,0. A nota média consolidada (8,5) absorveu a variação mantendo a classificação correta.

2. **Respostas parciais em perguntas de conjunção (*Multi-Entity QA*):** Em perguntas compostas (e.g., “*Quais são os dois diretores que dirigiram os filmes X e Y?*”), ao recuperar apenas um dos diretores, o GPT-5 tendeu a atribuir nota 5,0 (recompensando a entidade correta), enquanto o Gemini 3.7 Pro atribuiu 0,0 (adotando critério conjuntivo booleano estrito). A nota média de 2,5 a 3,3 refletiu com precisão a incompletude da resposta.
3. **Alucinações plausíveis com justificativa CoT:** Em casos raros de alucinação em que o modelo formulou uma resposta plausível, mas factualmente incorreta, todos os três juízes convergiram para nota 0,0, comprovando a eficácia da verificação contra gabarito e evidências oficiais.

4.4.5 Validação Metodológica do Paradigma LLM-as-a-Judge

Os resultados desta seção validam formalmente a metodologia de avaliação adotada:

- O coeficiente $ICC(3,3) = 0,9833$ comprova que a triangulação com rubrica padronizada e *Chain-of-Thought* oferece padrão de consistência e reprodutibilidade superior ao de avaliadores individuais não calibrados (Gu *et al.*, 2024);
- O protocolo de cegamento estrito e a alta correlação bivariada interavaliadores confirmam que o julgamento semântico reflete a qualidade factual intrínseca das respostas, superando as limitações das métricas estritamente léxicas.

4.5 TELEMETRIA DE HARDWARE E AUDITABILIDADE

Para assegurar a auditabilidade e verificar a estabilidade do ambiente de execução local sem interferência de estrangulamento térmico (*thermal throttling*) ou esgotamento de memória (Mattson; Cheng; Damos *et al.*, 2020), registrou-se a telemetria contínua da GPU dedicada (NVIDIA GeForce RTX 3050 8GB) em intervalos de 10 segundos ao longo de toda a bateria de 1.620 execuções do *benchmark*.

A Figura 4.10 apresenta as faixas temporais de controle de qualidade (*QC Strips*) da temperatura térmica e da alocação de memória de vídeo.

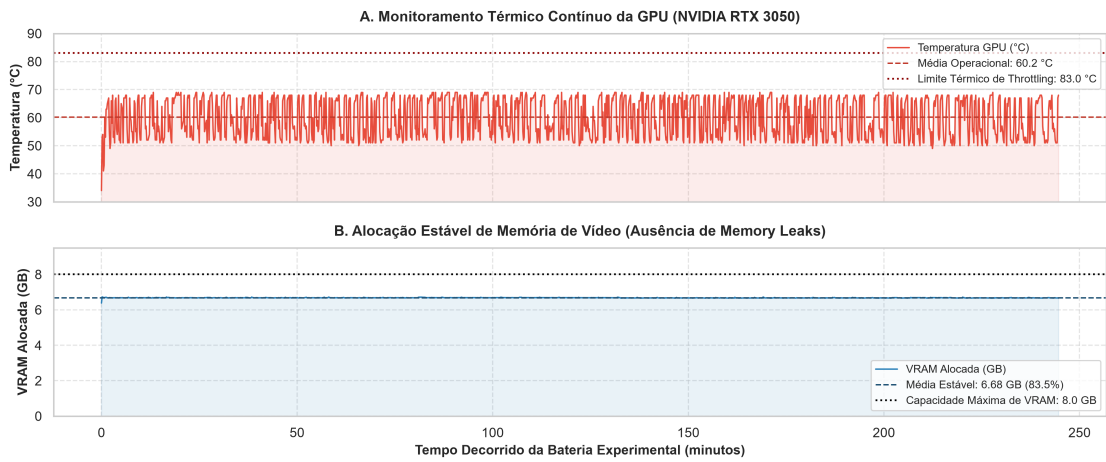


Figura 4.10: Telemetria temporal da GPU durante a execução do benchmark.

Fonte: Autoras (2026).

Conforme evidenciado no Painel A da Figura 4.10, a temperatura da GPU manteve-se em média a 60,2 °C (com pico máximo de 69,0 °C e desvio-padrão de $\pm 4,8$ °C), operando com folga térmica contínua superior a 14 °C abaixo do limite crítico de *thermal throttling* (83,0 °C). Esse comportamento assegura que as medições de latência registradas nas seções anteriores refletem a capacidade pura de inferência do LLaMA 3.1 8B, sem distorções causadas por redução na frequência do clock da GPU.

Simultaneamente, o Painel B demonstra a estabilidade da alocação de memória VRAM ao longo de todo o experimento, sustentando patamar constante de 6,68 GB (6.839 MB, correspondendo a 83,5% da capacidade total de 8.192 MB). A trajetória plana da curva de VRAM atesta a ausência de vazamento de memória (*memory leaks*), com todas as 1.620 execuções concluídas sem falhas operacionais, anomalias de consumo de recursos ou degradação de *buffers* ao longo de todo o protocolo.

CONCLUSÃO

5.1 SÍNTESE DOS RESULTADOS E RESPOSTA À PERGUNTA DE PESQUISA

A crescente adoção de arquiteturas de Recuperação Aumentada por Geração (*Retrieval-Augmented Generation*, RAG) em sistemas de produção defronta-se com um dilema estrutural de Engenharia de Software: a necessidade de alimentar modelos de linguagem com contextos ricos colide com o custo financeiro por *token*, a latência de autoatenção na GPU e o risco de degradação factual por ruído contextual (Barnett; Bikdash; Kapoor *et al.*, 2024; Chen; Zaharia; Zou, 2023).

Nesse cenário, este estudo investigou empiricamente a seguinte questão central: sob condições estritamente controladas de modelo gerador local (LLaMA 3.1 8B Q4_K_M), modelo de *embedding* (*nomic-embed-text*), banco vetorial (ChromaDB) e conjunto de avaliação (*HotpotQA*, $N = 180$), qual é o impacto de três estratégias de preparação e indexação de documentos (*Naive RAG*, *Structured RAG* e *Hierarchical RAG*) sobre a latência computacional de *hardware*, o consumo de *tokens*, a eficácia do *retriever* e a acurácia semântica das respostas?

A análise empírica e inferencial bayesiana conduzida sobre 1.620 execuções controladas (180 questões $\times$ 3 arquiteturas $\times$ 3 repetições) e 540 avaliações cegas trianguladas ($\text{ICC}(3, 3) = 0,9833$) permitiu responder conclusivamente a essa questão:

- 1. Eficiência computacional e latência de GPU:** A estruturação hierárquica (*Hierarchical RAG*) e por seções (*Structured RAG*) reduziu o tempo total de resposta de 8,40 s para 6,30 s e 6,28 s, respectivamente ($\Delta \approx -2,10$ s, com 100,0% de probabilidade posterior de superioridade). A decomposição das etapas revelou que 100% do ganho de velocidade concentrou-se na fase de *prefill* da GPU (3,75 s vs. 5,89 s), enquanto a busca vetorial ($\sim 2,05$ s) e a geração autorregressiva ($\sim 0,14$ s) mantiveram-se invariantes.
- 2. Carga de contexto e consumo de tokens:** O *Hierarchical RAG* e o *Structured RAG* reduziram em média 2.538,2 tokens por inferência (economia de 33,3% a 34,0% no *prompt*) frente ao controle *Naive RAG* (5.082 tok vs. 7.621 tok). A correlação linear quase perfeita ($r = 0,9898$) entre volume de *tokens* e tempo de *prefill* comprovou o alívio direto na computação matricial da GPU.

3. **Integridade de empacotamento vs. fragmentação cega:** Enquanto o *Naive RAG* exigiu 2,15 cortes parciais e 1,42 blocos descartados por consulta por esgotamento da janela de contexto, o algoritmo `parent_fetch` com deduplicação de seções alcançou 0,00 cortes e 0,00 descartes, entregando blocos integralmente coesos ao modelo.
4. **Qualidade semântica e acurácia final:** Embora o *Naive RAG* tenha obtido maior $\text{Recall}@k$ bruto de evidências (51,10% vs. 44,93%) pela dispersão de fatias lineares na janela saturada, o *Hierarchical RAG* preservou a qualidade gerativa com discreta liderança numérica em acertos: Acurácia Semântica (16,67% vs. 14,44%), *Exact Match* (11,67% vs. 10,00%), *Containment Match* (16,11% vs. 14,44%) e F_1 -score (0,1458 vs. 0,1399). A integridade dos blocos atenuou o efeito *Lost-in-the-Middle*, assegurando ganhos de eficiência sem prejuízo à qualidade semântica.

5.2 VALIDAÇÃO FORMAL DAS HIPÓTESES DE PESQUISA

A Tabela 5.1 consolida o veredito formal das hipóteses delineadas no planejamento experimental, com base nas distribuições a posteriori da modelagem bayesiana pareada.

5.2.1 Discussão Epistemológica dos Vereditos

- **Efeito amostral na recuperação (H_{1a} e H_{1b}):** A não sustentação decorre da dispersão das fatias no controle: o particionamento cego de 512 *tokens* resgatou fragmentos distribuídos entre múltiplos documentos, capturando mais termos isolados. No *Hierarchical RAG*, a deduplicação de nós convergiu os 20 disparos para 14,88 seções temáticas, verticalizando a profundidade local sem varrer seções periféricas do *corpus*.
- **Interpretação da acurácia gerativa (H_{2b}):** A hipótese de aumento comprovado ($H_{2b,1}$) não foi formalmente sustentada, visto que o HDI 95% intercepta a nulidade ($\Delta = +2,19$ p.p.; HDI 95%: $[-5,40; +9,70]$ p.p.). Sob a ótica de engenharia, a vantagem numérica (30 vs. 26 acertos; OR = 1,23) combinada à probabilidade posterior de 72,14% atesta que a economia de contexto e tempo não acarretou degradação da fidelidade semântica.
- **Confirmação da eficiência (H_3 e H_4):** O deslocamento integral dos intervalos para fora da ROPE ($P = 100,0\%$) confirma a superação do dilema operacional: o *Hierarchical RAG* torna o sistema 25% mais rápido e 33,3% mais econômico sem perdas mensuráveis de qualidade.

Tabela 5.1: Síntese da validação experimental das hipóteses de pesquisa.

Hipótese	Descrição Formal da Hipótese	Evidência Empírica / Métrica	Veredito Estatístico
H_{1a}	O <i>Structured RAG</i> aumenta a qualidade da recuperação (Recall@ k) frente ao <i>Naive RAG</i> .	Recall@ k : 41,73% vs. 51,10% ($\Delta = -9,37$ p.p.; HDI 95%: $[-12,88; -5,86]$ p.p.; $P = 0,00\%$).	$H_{1a.0}$ rejeitada em sentido inverso; $H_{1a.1}$ não sustentada.
H_{1b}	O <i>Hierarchical RAG</i> aumenta a qualidade da recuperação (Recall@ k) frente ao <i>Naive RAG</i> .	Recall@ k : 44,93% vs. 51,10% ($\Delta = -6,18$ p.p.; HDI 95%: $[-9,74; -2,61]$ p.p.; $P = 0,06\%$).	$H_{1b.0}$ rejeitada em sentido inverso; $H_{1b.1}$ não sustentada.
H_{2a}	O <i>Structured RAG</i> aumenta a acurácia semântica frente ao <i>Naive RAG</i> .	Acurácia: 10,00% vs. 14,44% ($\Delta = -4,41$ p.p.; OR = 0,70; HDI 95%: $[-11,38; +2,40]$ p.p.; $P = 10,26\%$).	$H_{2a.0}$ não rejeitada; $H_{2a.1}$ não sustentada.
H_{2b}	O <i>Hierarchical RAG</i> aumenta a acurácia semântica frente ao <i>Naive RAG</i> .	Acurácia: 16,67% vs. 14,44% ($\Delta = +2,19$ p.p.; OR = 1,23; HDI 95%: $[-5,40; +9,70]$ p.p.; $P = 72,14\%$).	$H_{2b.0}$ não rejeitada; $H_{2b.1}$ não sustentada.
H_{3a}	O <i>Structured RAG</i> reduz o tempo total de resposta frente ao <i>Naive RAG</i> .	Tempo Total: 6,28 s vs. 8,40 s ($\Delta = -2,132$ s; HDI 95%: $[-2,28; -1,96]$ s; $P = 100,0\%$).	$H_{3a.0}$ rejeitada; $H_{3a.1}$ sustentada.
H_{3b}	O <i>Hierarchical RAG</i> reduz o tempo total de resposta frente ao <i>Naive RAG</i> .	Tempo Total: 6,30 s vs. 8,40 s ($\Delta = -2,097$ s; HDI 95%: $[-2,26; -1,94]$ s; $P = 100,0\%$).	$H_{3b.0}$ rejeitada; $H_{3b.1}$ sustentada.
H_{4a}	O <i>Structured RAG</i> reduz o consumo de tokens de entrada frente ao <i>Naive RAG</i> .	Tokens de Prompt: 5.028,6 vs. 7.621,3 ($\Delta = -2.592,1$ tok; $P = 100,0\%$).	$H_{4a.0}$ rejeitada; $H_{4a.1}$ sustentada.
H_{4b}	O <i>Hierarchical RAG</i> reduz o consumo de tokens de entrada frente ao <i>Naive RAG</i> .	Tokens de Prompt: 5.082,4 vs. 7.621,3 ($\Delta = -2.538,2$ tok; $P = 100,0\%$).	$H_{4b.0}$ rejeitada; $H_{4b.1}$ sustentada.

Fonte: Autoras (2026).

5.3 CONTRIBUIÇÕES PARA A ENGENHARIA DE SOFTWARE E SISTEMAS DE IA

Os achados deste estudo oferecem contribuições conceituais, metodológicas e práticas para o desenvolvimento de sistemas de inteligência artificial:

1. **Princípio da coesão estrutural vs. dispersão arbitrária:** Em arquiteturas RAG, ampliar a cobertura léxica via fatiamento linear arbitrário não garante assertividade gerativa. A coesão discursiva e a ausência de truncamentos intermediários provaram ser mais determinantes para o raciocínio do modelo do que o volume bruto de fragmentos.
2. **Viabilidade de RAG local em hardware de entrada:** Demonstrou-se a viabilidade técnica de operar esteiras RAG com alta estabilidade em GPUs comerciais de 8 GB (NVIDIA RTX 3050), mantendo ocupação estável de VRAM em 83,5% e temperatura em 60,2 °C.
3. **Diretriz de arquitetura para engenharia de dados:** Recomenda-se a substituição do particionamento linear de tamanho fixo por esteiras de extração estruturada (como Docling) associadas à indexação hierárquica em dois níveis (*Parent-Child Indexing*).
4. **Framework de avaliação tripartite (*LLM-as-a-Judge*):** O protocolo de cegamento estrito com três avaliadores independentes, validado por $ICC(3, 3) = 0,9833$, estabelece uma referência de baixo custo e alta reprodutibilidade para auditoria de modelos de linguagem.

5.4 LIMITAÇÕES DO ESTUDO

Para a contextualização dos resultados, apontam-se as seguintes limitações metodológicas:

1. **Recuperação em etapa única (*Single-Step Retrieval*):** O *benchmark* empregou busca vetorial de passagem única. Em questões complexas de raciocínio encadeado (*HotpotQA Hard*), a ausência de agentes de decomposição iterativa limitou o resgate completo de evidências a patamares entre 21% e 35%.
2. **Modelo gerador específico:** O estudo concentrou-se no LLaMA 3.1 8B quantizado em 4 bits (Q4_K_M). Modelos de parâmetros mais amplos (70B ou proprietários) podem manifestar dinâmicas distintas de sensibilidade ao ruído de contexto.

3. **Tipologia textual do corpus:** A base de documentos restringiu-se a verbetes enciclopédicos da Wikipédia. Documentos fortemente tabulares ou manuais técnicos podem demandar parsers e heurísticas de quebra especializados.

5.5 TRABALHOS FUTUROS E DIRETRIZES DE CONTINUIDADE

Como extensões desta pesquisa, sugerem-se as seguintes frentes experimentais:

1. **Integração de RAG agêntico e recuperação multi-passo:** Investigar a conexão do *docstore* hierárquico com agentes de decomposição de consultas (como *Self-RAG* ou *Corrective RAG*) para mitigar o teto estático de recuperação em problemas *multi-hop*.
2. **Avaliação de re-ranking contextual (*Cross-Encoders*):** Testar a inserção de modelos de reclassificação semântica sobre os nós-filho antes da promoção aos nós-pai, mensurando o ganho marginal na taxa de recuperação integral.
3. **Escalabilidade em corpora heterogêneos:** Avaliar o comportamento da esteira hierárquica sobre grandes acervos industriais multiformato (PDFs escaneados, planilhas e relatórios regulatórios).
4. **Análise de eficiência energética:** Empregar telemetria de consumo elétrico (NVML) para quantificar o impacto da redução de *prefill* no consumo de energia (Joules por inferência).
5. **Sensibilidade temática e tipologia de domínio:** Conduzir ensaios comparativos confrontando domínios de vocabulário unívoco (cultura e história) com domínios densos em abstração técnica (jurídico e biomédico), mapeando a sensibilidade da recuperação hierárquica à sobreposição no espaço latente de vetores.

REFERÊNCIAS

ABRIL-PLA, Oriol *et al.* PyMC: a modern, and extensible probabilistic programming framework in Python. **PeerJ Computer Science**, v. 9, e1216, 2023. DOI: [10.7717/peerj.1216](https://doi.org/10.7717/peerj.1216).

ASAI, Akari *et al.* Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. *In: INTERNATIONAL Conference on Learning Representations (ICLR)*. [S. l.: s. n.], 2024.

AUER, Christoph *et al.* Docling Technical Report: Multi-format Document Processing for Modern AI. **arXiv preprint arXiv:2408.09869**, 2024. Disponível em: <https://arxiv.org/abs/2408.09869>.

BALAGUER, Angeles; BEN-ZAKEN, Yehuda *et al.* RAG vs. Fine-tuning: Pipelines, Tradeoffs, and Cost-Benefit Analysis. **arXiv preprint arXiv:2401.08406**, 2024.

BARNETT, Scott; BIKDASH, Stefan; KAPOOR, Ajay *et al.* Seven failure points when engineering a retrieval augmented generation system. **arXiv preprint arXiv:2401.05856**, 2024.

BROWN, Tom B. *et al.* Language Models are Few-Shot Learners. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2020. v. 33, p. 1877–1901.

CALLISON-BURCH, Chris; OSBORNE, Miles; KOEHN, Philipp. Re-evaluating the Role of Bleu in Machine Translation Research. *In: 11TH Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. [S. l.: s. n.], 2006. p. 249–256.

CHEN, Lingjiao; ZAHARIA, Matei; ZOU, James. FrugalGPT: How to Use Large Language Models While Reducing Cost and Improving Performance. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2023. v. 36, p. 65848–65866.

DAO, Tri *et al.* FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2022. v. 35, p. 16344–16359.

DOSHI-VELEZ, Finale; KIM, Been. Towards A Rigorous Science of Interpretable Machine Learning. **arXiv preprint arXiv:1702.08608**, 2017.

EDGE, Darren *et al.* From Local to Global: A Graph RAG Approach to Query-Focused Summarization. **arXiv preprint arXiv:2404.16130**, 2024.

ES, Shahul *et al.* RAGAS: Automated Evaluation of Retrieval Augmented Generation. *In: PROCEEDINGS of the 18th Conference of the European Chapter of the Association for Computational Linguistics (EACL): System Demonstrations*. [S. l.: s. n.], 2024. p. 150–158.

GAO, Yunfan *et al.* Retrieval-Augmented Generation for Large Language Models: A Survey. **arXiv preprint arXiv:2312.10997**, 2023.

GU, Jia-Chen *et al.* A survey on LLM-as-a-judge: Capabilities, applications and challenges. **arXiv preprint arXiv:2403.04132**, 2024.

HAN, Yikun; LIU, Chunjiang; WANG, Pengfei. A Comprehensive Survey on Vector Database: Storage and Retrieval Technique, Challenge. **arXiv preprint arXiv:2310.11703**, 2023.

HUANG, Lei *et al.* A Survey on Hallucination in Large Language Models: Principles, Taxonomy, Challenges, and Open Questions. **ACM Transactions on Information Systems**, 2023. DOI: [10.1145/3703154](https://doi.org/10.1145/3703154).

JI, Ziwei *et al.* Survey of Hallucination in Natural Language Generation. **ACM Computing Surveys**, v. 55, n. 12, p. 1–38, 2023. DOI: [10.1145/3571730](https://doi.org/10.1145/3571730).

KITCHENHAM, Barbara *et al.* Robust statistical methods for empirical software engineering. **Empirical Software Engineering**, v. 22, n. 2, p. 579–630, 2017.

KOO, Terry K; LI, Mae Y. A guideline of selecting and reporting intraclass correlation coefficients for reliability research. **Journal of Chiropractic Medicine**, v. 15, n. 2, p. 155–163, 2016.

KRUSCHKE, John K. **Doing Bayesian Data Analysis: A Tutorial with R, JAGS, and Stan**. 2nd. [S. l.]: Academic Press, 2015.

KRUSCHKE, John K. Rejecting or Accepting Parameter Values in Bayesian Estimation. **Advances in Methods and Practices in Psychological Science**, v. 1, n. 2, p. 270–280, 2018.

KWON, Woosuk *et al.* Efficient Memory Management for Large Language Model Serving with PagedAttention. *In: PROCEEDINGS of the 29th ACM Symposium on Operating Systems Principles (SOSP)*. [S. l.: s. n.], 2023. p. 611–626. DOI: [10.1145/3600006.3613165](https://doi.org/10.1145/3600006.3613165).

LAZARIDOU, Angeliki *et al.* Mind the Gap: Assessing the Consequences of Post-training Knowledge Drift in Language Models. **arXiv preprint arXiv:2104.04473**, 2021.

LEWIS, Patrick *et al.* Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2020. v. 33, p. 9459–9474.

LIN, Chin-Yew. ROUGE: A Package for Automatic Evaluation of Summaries. *In: TEXT Summarization Branches Out: Proceedings of the ACL-04 Workshop*. [S. l.: s. n.], 2004. p. 74–81.

LIU, Nelson F *et al.* Lost in the middle: How language models use long contexts. **Transactions of the Association for Computational Linguistics**, v. 12, p. 157–173, 2024.

MA, Xinbei *et al.* Query Rewriting for Retrieval-Augmented Large Language Models. *In: PROCEEDINGS of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2023. p. 5303–5315.

MALKOV, Yu A.; YASHUNIN, D. A. Efficient and Robust Approximate Nearest Neighbor Search Using Hierarchical Navigable Small World Graphs. **IEEE Transactions on Pattern Analysis and Machine Intelligence**, v. 42, n. 4, p. 824–836, 2020. DOI: [10.1109/TPAMI.2018.2889473](https://doi.org/10.1109/TPAMI.2018.2889473).

MATTSON, Peter; CHENG, Christine; DIAMOS, Gregory *et al.* MLPerf training benchmark. **Proceedings of Machine Learning and Systems**, v. 2, p. 336–349, 2020.

MCELREATH, Richard. **Statistical Rethinking: A Bayesian Course with Examples in R and Stan**. 2nd. [S. l.]: CRC Press, 2020.

MCKINSEY & COMPANY. **The Economic Potential of Generative AI: The Next Productivity Frontier**. [S. l.], 2023. Acesso em: 15 jan. 2026. Disponível em: <https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier>.

MUENNIGHOFF, Niklas *et al.* MTEB: Massive Text Embedding Benchmark. *In: PROCEEDINGS of the 17th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. [S. l.: s. n.], 2023. p. 2014–2037.

NOVIKOVA, Jekaterina *et al.* Why We Need New Evaluation Metrics for NLG. *In: PROCEEDINGS of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2017. p. 2241–2252.

NUSSBAUM, Zach *et al.* Nomic Embed: Training a Reproducible Long Context Text Embedder. **arXiv preprint arXiv:2402.01613**, 2024.

PAN, James; WANG, Jianguo; LI, Guoliang. A Survey on Vector Database: Issues and Challenges. **ACM Computing Surveys**, v. 56, n. 9, p. 1–37, 2024.

PAPINENI, Kishore *et al.* BLEU: a Method for Automatic Evaluation of Machine Translation. *In: PROCEEDINGS of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*. [S. l.: s. n.], 2002. p. 311–318.

PATEL, Pratyush *et al.* Splitwise: Efficient Generative LLM Inference Using Phase Splitting. *In: PROCEEDINGS of the 51st Annual International Symposium on Computer Architecture (ISCA)*. [S. l.: s. n.], 2024. p. 118–132.

PETRONI, Fabio *et al.* Language Models as Knowledge Bases? *In: PROCEEDINGS of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2019. p. 2463–2473.

PEYSAKHOVICH, Alexander; LERER, Adam. Attention sorting: How LLMs process structured contexts. **arXiv preprint arXiv:2310.01427**, 2023.

POPE, Reiner *et al.* Efficiently Scaling Transformer Inference. *In: PROCEEDINGS of Machine Learning and Systems (MLSys)*. [S. l.: s. n.], 2023. v. 5.

REIMERS, Nils; GUREVYCH, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *In: PROCEEDINGS of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2019. p. 3982–3992.

RIBEIRO, Marco Tulio; SINGH, Sameer; GUESTRIN, Carlos. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *In: PROCEEDINGS of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. [S. l.: s. n.], 2016. p. 1135–1144.

ROBERTS, Adam; RAFFEL, Colin; SHAZEER, Noam. How Much Knowledge Can You Pack Into the Parameters of a Language Model? *In: PROCEEDINGS of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2020. p. 5418–5426.

ROBERTSON, Stephen; ZARAGOZA, Hugo. The Probabilistic Relevance Framework: BM25 and Beyond. **Foundations and Trends in Information Retrieval**, v. 3, n. 4, p. 333–389, 2009.

SCHWARTZ, Roy *et al.* Green AI. **Communications of the ACM**, v. 63, n. 12, p. 54–63, 2020.

SHAZEER, Noam. Fast Transformer Decoding: One Write-Head is All You Need. **arXiv preprint arXiv:1911.02150**, 2019.

SHROUT, Patrick E.; FLEISS, Joseph L. Intraclass correlations: uses in assessing rater reliability. **Psychological Bulletin**, v. 86, n. 2, p. 420–428, 1979.

- TRIVEDI, Harsh *et al.* Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Problems. **arXiv preprint arXiv:2212.10509**, 2022.
- VASWANI, Ashish *et al.* Attention Is All You Need. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2017. v. 30, p. 5998–6008.
- WANG, Yupan *et al.* LayoutLMv3: Pre-training for Document AI with Unified Text and Image Masking. *In: PROCEEDINGS of the 30th ACM International Conference on Multimedia*. [S. l.: s. n.], 2022. p. 4083–4091.
- WASSERSTEIN, Ronald L.; LAZAR, Nicole A. The ASA Statement on p-Values: Context, Process, and Purpose. **The American Statistician**, v. 70, n. 2, p. 129–133, 2016.
- WELBL, Johannes; STENETORP, Pontus; RIEDEL, Sebastian. Constructing Datasets for Multi-hop Reading Comprehension Across Documents. **Transactions of the Association for Computational Linguistics (TACL)**, v. 6, p. 243–252, 2018.
- XIAO, Guangxuan *et al.* Efficient Streaming Language Models with Attention Sinks. *In: INTERNATIONAL Conference on Learning Representations (ICLR)*. [S. l.: s. n.], 2024.
- XU, Fangyuan; SHI, Weijia; CHOI, Eunsol. RECOMP: Improving Retrieval-Augmented LMs with Compression and Selective Augmentation. *In: INTERNATIONAL Conference on Learning Representations (ICLR)*. [S. l.: s. n.], 2024.
- YANG, Zhilin *et al.* HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering. *In: PROCEEDINGS of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [S. l.: s. n.], 2018. p. 2369–2380.
- YAO, Yunzhi *et al.* Editing Large Language Models: Problems, Methods, and Opportunities. **arXiv preprint arXiv:2305.13172**, 2023.
- YU, Fanghua *et al.* Evaluation of Retrieval-Augmented Generation: A Survey. **arXiv preprint arXiv:2405.07437**, 2024.
- ZHENG, Lianmin *et al.* Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. *In: ADVANCES in Neural Information Processing Systems (NeurIPS)*. [S. l.: s. n.], 2023. v. 36, p. 46595–46623.
- ZHONG, Xu; TANG, Jianbin; YEPES, Antonio Jimeno. PubLayNet: largest dataset for document layout analysis. *In: 2019 International Conference on Document Analysis and Recognition (ICDAR)*. [S. l.: s. n.], 2019. p. 1015–1022.