

**EMBRAPII COMPETENCE CENTER IN IMMERSIVE
TECHNOLOGIES (AKCIT)**

**POSTGRADUATE TRAINING PROGRAM IN IMMERSIVE
TECHNOLOGIES AND ARTIFICIAL INTELLIGENCE**

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

**IMPACT OF CORPUS PREPARATION AND
RETRIEVAL ON THE EFFICIENCY OF RAG
SYSTEMS:**

A PAIRED BENCHMARK ON MULTI-HOP QUESTIONS

Goiânia – GO, Brazil

2026

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

**IMPACT OF CORPUS PREPARATION AND
RETRIEVAL ON THE EFFICIENCY OF RAG
SYSTEMS:**

A PAIRED BENCHMARK ON MULTI-HOP QUESTIONS

Final Capstone Project submitted to the Training and Education Program of the Embraer Competence Center in Immersive Technologies (AKCIT) as a mandatory requirement for obtaining the certificate of specialization in Artificial Intelligence and Natural Language Processing.

Goiânia – GO, Brazil

2026

LARISSA NEPOMUCENO COSTA MARINHO
VANESSA SCHEMES MARTINS PINTO

IMPACT OF CORPUS PREPARATION AND
RETRIEVAL ON THE EFFICIENCY OF RAG
SYSTEMS:
A PAIRED BENCHMARK ON MULTI-HOP QUESTIONS

Final Capstone Project submitted to the
Training and Education Program of the Em-
brapii Competence Center in Immersive Tech-
nologies (AKCIT) as a mandatory require-
ment for obtaining the certificate of special-
ization in Artificial Intelligence and Natural
Language Processing.

Approved on: September 23, 2026.

EXAMINATION BOARD

Prof. Dr. [Advisor Name] – Advisor / Chair
Embrapii Competence Center in Immersive Technologies (AKCIT)

Prof. Dr. [Internal Committee Member] – Internal Member
Embrapii Competence Center in Immersive Technologies (AKCIT)

Prof. Dr. [External Committee Member] – External Member
Guest University / Research Institute

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 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.

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.

LIST OF FIGURES

3.1	Repository directory layout and modular pipeline architecture.	21
3.2	Experimental pipeline data flow and intermediate artifact traceability. . . .	22
3.3	Architectural comparison of the three evaluated RAG scenarios.	24
4.1	Comparative summary of overall performance across RAG scenarios.	39
4.2	Empirical distribution of total response latency across RAG architectures. .	40
4.3	Posterior 95% highest density intervals (HDI) and ROPE for paired latency contrasts.	42
4.4	Mean latency decomposition across RAG pipeline stages.	43
4.5	Posterior Bayesian contrasts across pipeline stages (Prefill, Vector Retrieval, and Generation) with 95% HDI and ROPE.	44
4.6	Forest plot of 95% highest density intervals (HDI) and ROPE for prompt token differences.	47
4.7	Mean prompt token consumption (Panel A) and linear correlation between token volume and GPU prefill latency (Panel B).	48
4.8	Supporting evidence retrieval efficacy (Panel A) and semantic accuracy across difficulty tiers (Panel B).	53
4.9	Triangulated judge panel metrics: correlation matrix (Panel A), bimodal score density (Panel B), and continuous mean scores by RAG architecture (Panel C).	56
4.10	GPU hardware telemetry throughout benchmark execution.	58

LIST OF TABLES

3.1	Software ecosystem, libraries, and experimental architecture components. . .	30
4.1	Descriptive statistics of total inference latency across RAG architectures ($N = 180$).	40
4.2	Posterior estimates of paired latency differences (Δ Time in seconds). . . .	41
4.3	Mean latency decomposition across RAG pipeline stages ($N = 180$).	42
4.4	Descriptive metrics of input token consumption and packing efficiency ($N = 180$).	45
4.5	Posterior estimates of paired prompt token differences (Δ Prompt Tokens). .	46
4.6	Global retrieval efficacy and generation quality metrics ($N = 180$).	49
4.7	Posterior estimates of retrieval hit rates and generative accuracy.	50
4.8	Retrieval efficacy and generative accuracy stratified by difficulty level ($n =$ 60 per tier).	52
4.9	Inter-rater reliability and agreement matrix across judges (N = 540 evaluated answers)	55
4.10	Pairwise Pearson (r) and Spearman (r_s) correlations across triangulated judges.	55
4.11	Descriptive statistics of judge scores and consolidated averages (0 to 10 points). .	56
5.1	Summary of experimental research hypothesis validation.	61

LIST OF ABBREVIATIONS AND ACRONYMS

AKCIT	Embrapii Competence Center in Immersive Technologies
API	Application Programming Interface
BPE	Byte-Pair Encoding
CoT	Chain-of-Thought
CPU	Central Processing Unit
CUDA	Compute Unified Device Architecture
EM	Exact Match
EMBRAPII	Brazilian Research and Industrial Innovation Company
ESS	Effective Sample Size
GPU	Graphics Processing Unit
HDI	Highest Density Interval
HNSW	Hierarchical Navigable Small World
HTML	HyperText Markup Language
ICC	Intraclass Correlation Coefficient
IQR	Interquartile Range
KV-Cache	Key-Value Cache
LLM	Large Language Model
MCMC	Markov Chain Monte Carlo
NLG	Natural Language Generation
NLP	Natural Language Processing
NUTS	No-U-Turn Sampler
OCR	Optical Character Recognition
OPEX	Operational Expenditure
OR	Odds Ratio
RAG	Retrieval-Augmented Generation
ROPE	Region of Practical Equivalence
VRAM	Video Random Access Memory

CONTENTS

1	Introduction	7
1.1	Research Objectives	8
1.1.1	General Objective	8
1.1.2	Specific Objectives	8
1.2	Research Question and Working Hypotheses	9
1.3	Contributions and Dissertation Outline	10
2	Theoretical Foundations and Related Work	12
2.1	Autoregressive Neural Language Models and Attention Dynamics	12
2.2	The RAG Paradigm and Non-Parametric Memory Modeling	13
2.3	Document Layout Analysis and Text Structural Engineering	14
2.4	Dense Vector Indexing and Approximate Nearest Neighbor Search	15
2.5	Multi-Hop Reasoning and Contextual Degradation (Lost in the Middle) . .	16
2.6	Triangulated Semantic Evaluation and Bayesian Statistical Inference	17
3	Experimental Methodology	20
3.1	Experimental Architecture and Evaluated RAG Scenarios	20
3.1.1	Overview and Modular Software Architecture	20
3.1.2	End-to-End Data Flow	21
3.1.3	Characterization of Evaluated RAG Scenarios and Ablation Design	22
3.1.4	Hierarchical Parent Section Resolution	24
3.1.5	Context Window Management and Dynamic Token Budgeting . . .	24
3.2	Stratified Sampling and Query Selection	25
3.2.1	Data Source and Domain Scope	25
3.2.2	Difficulty Stratification and Deterministic Sampling	26
3.2.3	Human Validation and 1:1 Replacement Protocol	26
3.3	Corpus Construction, Sanitization, and Integrity Auditing	27

3.3.1	Historical Freezing and Data Drift Mitigation	27
3.3.2	DOM Pre-Sanitization and Format Parity	28
3.3.3	Automated Corpus Audit and Empirical Coverage Bounds	28
3.4	Operational Environment, Hardware, and Software Infrastructure	28
3.4.1	Hardware Specifications and Experimental Isolation	28
3.4.2	Language Models, Embeddings, and Parameter Configuration	29
3.5	Paired Execution Protocol, Randomization, and Bias Controls	31
3.5.1	Paired Experimental Design and Sample Size	31
3.5.2	Order Randomization and Interleaved Global Rounds	31
3.5.3	Strict Serial Concurrency Regime ($N = 1$)	31
3.5.4	Warm-Up Protocol and Cache Cold-Start Elimination	32
3.5.5	Continuous Background Infrastructure Telemetry	32
3.6	Performance Indicators, Telemetry Metrics, and Data Mapping	32
3.6.1	Context Retrieval Efficacy Metrics	32
3.6.2	Secondary Lexical Generation Quality Metrics (Descriptive Analysis)	33
3.6.3	Latency Telemetry and Execution Phase Decomposition	33
3.6.4	Context Allocation and Computational Cost Metrics	34
3.6.5	Master Dataset Consolidation and Artifact Layout	34
3.7	Double-Blind Evaluation Protocol and Semantic Quality Validation	35
3.7.1	Overcoming Lexical Limitations via Semantic Judging and H_2 Formalization	35
3.7.2	Blinding Protocol and Anonymization	35
3.7.3	Tripartite Judge Triangulation Panel	36
3.7.4	Inter-Rater Reliability and Methodological Safeguards	36
3.7.5	Bayesian Inferential Modeling and Hypothesis Testing	37
3.8	Reproducibility, Auditability, and Ethical Considerations	37
3.8.1	Statement on the Use of AI Assistance in Code Implementation	38
4	Results and Discussion	39
4.1	COMPARATIVE ANALYSIS OF RESPONSE LATENCY AND HARDWARE EFFICIENCY	39

4.1.1	Global Latency Performance: Cross-Architectural Comparison . . .	39
4.1.2	Bayesian Modeling of Paired Response Latency	40
4.1.3	Latency Decomposition by Pipeline Stage: Root Cause of Speedup .	41
4.1.4	Latency Discussion Summary	44
4.2	ANALYSIS OF TOKEN CONSUMPTION, CONTEXT LOAD, AND PACKING EFFICIENCY	45
4.2.1	Prompt Token Load Across Architectures	45
4.2.2	Bayesian Modeling of Paired Token Consumption	46
4.2.3	Document Fragmentation and Parent Deduplication Dynamics . . .	46
4.2.4	Causal Relationship Between Token Load and Prefill Latency . . .	47
4.2.5	Mitigation of the Lost-in-the-Middle Effect and Cost Implications .	48
4.3	RETRIEVAL EFFICACY, SEMANTIC ACCURACY, AND COMPLEX- ITY STRATIFICATION	49
4.3.1	Global Retrieval and Generation Performance	49
4.3.2	Bayesian Logistic Modeling of Hit Rate and Semantic Accuracy . .	50
4.3.3	The Retrieval Paradox: Noisy Fragments versus Contextual Cohesion	50
4.3.4	Performance Stratification by Complexity Level (Easy, Medium, Hard)	52
4.3.5	Limitations of Single-Step Retrieval in Multi-Hop Reasoning	53
4.4	METHODOLOGICAL RELIABILITY AND JUDGE TRIANGULATION	54
4.4.1	Blind Evaluation Protocol and Rubric Standardization	54
4.4.2	Inter-Rater Reliability and Intraclass Correlation Coefficient (ICC)	55
4.4.3	Continuous Score Distributions and Score Tier Analysis	55
4.4.4	Qualitative Analysis of Edge Cases and Discrepancies	57
4.4.5	Methodological Validation of LLM-as-a-Judge	57
4.5	HARDWARE TELEMETRY AND AUDITABILITY	57
5	Conclusion	59
5.1	SYNTHESIS OF FINDINGS AND RESEARCH QUESTION RESOLUTION	59
5.2	FORMAL VALIDATION OF RESEARCH HYPOTHESES	60
5.2.1	Epistemological Discussion of Hypothesis Verdicts	60
5.3	CONTRIBUTIONS TO SOFTWARE ENGINEERING AND AI SYSTEMS	62

5.4	LIMITATIONS OF THE STUDY	62
5.5	FUTURE WORK AND RESEARCH DIRECTIONS	63

CHAPTER 1

INTRODUCTION

The recent convergence of Natural Language Processing (NLP) around Large Language Models (LLMs) has consolidated the widespread adoption of autoregressive neural architectures grounded in self-attention mechanisms (Vaswani *et al.*, 2017; Brown *et al.*, 2020). Despite the remarkable capabilities demonstrated by these models across complex linguistic tasks, their deployment in production software systems faces fundamental constraints inherent to parametric memory, notably factual hallucinations (Ji *et al.*, 2023; Huang *et al.*, 2023) and temporal obsolescence resulting from the static nature of model weights post-pre-training.

To alleviate exclusive reliance on parametric memory, the Retrieval-Augmented Generation (RAG) paradigm has become the dominant architectural solution (Lewis *et al.*, 2020). By decoupling the generative model from an external, dynamic knowledge base, RAG dynamically retrieves relevant document passages and injects them directly into the context prompt at inference time, thereby grounding model outputs in verifiable factual evidence without necessitating costly model retraining or fine-tuning.

However, operating RAG systems in production environments introduces critical software engineering tradeoffs and computational infrastructure challenges:

1. Severe processing overhead during the GPU prompt prefill phase, where ingesting extensive context windows, computing dense self-attention projection matrices, and allocating the Key-Value (KV) cache constitute the primary execution bottleneck of inference (Dao *et al.*, 2022; Kwon *et al.*, 2023);
2. Significant recurrent operational expenditure (OPEX) driven by the systematic transmission of redundant text passages, which inflates per-query costs in commercial API services and quickly saturates available Video RAM (VRAM) in local on-premise hardware (Balaguer; Ben-Zaken, *et al.*, 2024; Chen; Zaharia; Zou, 2023);
3. Generative fidelity degradation induced by attentional dispersion and context dilution (the Lost-in-the-Middle effect), caused by naive linear chunking strategies that arbitrarily fracture paragraph, list, and table boundaries, dispersing vital evidence across noisy and fragmented blocks (Liu *et al.*, 2024; Peysakhovich; Lerer, 2023).

In enterprise and embedded production environments, the practical viability of RAG systems is strictly bounded by generation correctness, strict latency budgets, and

compute resource constraints. While recent literature has explored increasingly intricate architectures, such as Knowledge Graph RAG (Graph RAG) (Edge *et al.*, 2024) or multi-agent recursive retrieval loops (Gao *et al.*, 2023; Asai *et al.*, 2024), these approaches introduce substantial asynchronous indexing overhead and multi-step inference delays. Conversely, the empirical impact of foundational interventions applied directly during document parsing, layout extraction, and corpus preparation remains insufficiently quantified under rigorous, hardware-controlled benchmarks.

Evaluating document preparation using modern layout-aware parsers, such as Docling (Auer *et al.*, 2024), combined with two-tier hierarchical retrieval (while strictly freezing the underlying hardware, generator model, embedding model, and query distribution), enables direct, unconfounded quantification of the fundamental tradeoffs governing GPU processing latency, token volumetric consumption, and semantic answer fidelity. Reducing context token volume without sacrificing evidence cohesion alleviates GPU attention load, directly facilitating stable, low-latency RAG deployment on consumer-grade local hardware.

1.1 RESEARCH OBJECTIVES

1.1.1 General Objective

To quantitatively and comparatively evaluate, through a strictly controlled experimental benchmark, the impact of three corpus preparation and retrieval architectures (Naive RAG, Structured RAG, and Hierarchical RAG) on retrieval efficacy, generative answer quality, computational hardware latency, and token context consumption, while holding strictly constant the local generative language model (Llama 3.1 8B quantized to 4 bits), the dense embedding model (nomic-embed-text), the vector database (ChromaDB), and the evaluation query set.

1.1.2 Specific Objectives

1. Extract and programmatically audit 180 multi-hop reasoning encyclopedic questions from the official HotpotQA benchmark, uniformly stratified across native complexity tiers (60 Easy, 60 Medium, and 60 Hard questions);
2. Construct and historically freeze a closed knowledge corpus comprising 1,780 full-text Wikipedia articles indexed prior to the dataset cutoff date (October 1, 2017), generating parallel representations in raw sanitized plain text (Group 1) and hierarchical structural Markdown via Docling (Groups 2 and 3);

3. Implement isolated, airtight vector collections in ChromaDB, benchmarking linear fixed-size chunking with 512-token windows (Group 1, Control), section-delimited structural parsing (Group 2), and a two-tier hierarchical architecture searching across 128-token child nodes with deterministic parent section retrieval and deduplication (Group 3);
4. Execute 1,620 serialized, paired inference trials under unitary concurrency ($N = 1$) across three technical repetitions (180 questions $\times$ 3 architectures $\times$ 3 repetitions), recording continuous GPU thermal telemetry and granular latencies decomposed into vector retrieval, prompt prefill, and autoregressive decoding;
5. Submit all 540 unique generated answers to a double-blind, tripartite evaluation protocol (comprising a human reference auditor and frontier models GPT-5 and Gemini 3.7 Pro), quantifying inter-rater reliability via the Intraclass Correlation Coefficient ICC(3, 3);
6. Estimate pairwise posterior contrast distributions via robust Student- t and Bayesian logistic regression modeling, establishing 95% Highest Density Intervals (HDI 95%), posterior probabilities of superiority, and formal evaluation against *a priori* declared Regions of Practical Equivalence (ROPE).

1.2 RESEARCH QUESTION AND WORKING HYPOTHESES

This research is guided by the following central question:

Under strictly controlled conditions of a local generative model (Llama 3.1 8B Q4_K_M), a dense embedding model (nomic-embed-text), a vector database (ChromaDB), and a multi-hop evaluation benchmark (HotpotQA, $N = 180$), what is the empirical impact of three corpus preparation and retrieval strategies (Naive RAG, Structured RAG, and Hierarchical RAG) on retrieval efficacy, generative answer quality, hardware computational latency, and prompt token consumption?

Based on the premise that preserving syntactic structure and optimizing retrieval granularity directly govern contextual information density, eight formal hypotheses were evaluated against the linear control group (Naive RAG, G1):

- H_1 (Evidence Retrieval Efficacy): Structural section chunking in Structured RAG (H_{1a}) and two-tier hierarchical retrieval in Hierarchical RAG (H_{1b}) increase the recovery rate of official supporting facts (Recall@20) relative to Naive RAG;

- H_2 (Generative Answer Quality): Structured RAG (H_{2a}) and Hierarchical RAG (H_{2b}) increase the semantic accuracy of model answers evaluated by the blind tripartite panel relative to Naive RAG;
- H_3 (Computational Execution Latency): Structured RAG (H_{3a}) and Hierarchical RAG (H_{3b}) reduce total end-to-end inference latency relative to Naive RAG;
- H_4 (Context Load and Token Consumption): Structured RAG (H_{4a}) and Hierarchical RAG (H_{4b}) reduce the volume of input prompt tokens injected into the language model relative to Naive RAG.

Decision rules combine 95% Highest Density Intervals (HDI 95%) with declared Regions of Practical Equivalence (ROPE) for continuous physical variables, and posterior probabilities of superiority for bounded discrete indicators.

1.3 CONTRIBUTIONS AND DISSERTATION OUTLINE

The primary contributions of this dissertation to Software Engineering and Natural Language Processing include:

1. **Decomposition and root-cause attribution of latency in RAG pipelines:** Empirical proof that the entire execution speedup delivered by structural architectures originates in the prompt prefill phase on the GPU (3.75 s in Hierarchical RAG versus 5.89 s in Naive RAG), while graph-based vector search and autoregressive decoding remain virtually invariant;
2. **Token volumetric efficiency without semantic degradation:** Demonstration that two-tier hierarchical parent-child retrieval achieves a 33.3% reduction in prompt token consumption compared to linear chunking (5,082 versus 7,621 tokens), while sustaining parity in generative answer accuracy (16.67% versus 14.44% semantic accuracy, and equivalent lexical F_1 scores of 0.1458 versus 0.1399);
3. **Syntactic boundary preservation and mitigation of greedy truncation:** Proof that retrieving complete parent sections with automatic deduplication eliminates textual fragmentation caused by greedy prompt packing, recording zero partial truncations and zero dropped nodes, contrasted with an average of 2.15 partial cuts and 1.42 dropped nodes per query in Naive RAG;
4. **Open, reproducible, and fully auditable benchmarking pipeline:** Complete public release of the experimental codebase, ChromaDB vector collections, relational SQLite artifacts, GPU thermal logs, and double-blind scoring matrices under open science principles.

This dissertation is structured across five chapters. Chapter 1 introduces the research problem, motivation, formal objectives, and experimental hypotheses. Chapter 2 surveys theoretical foundations, reviewing autoregressive models, self-attention mechanics, RAG pipelines, layout-aware document parsing, and modern LLM-as-a-Judge evaluation protocols.

Chapter 3 details the experimental methodology, specifying dedicated hardware isolation, historical corpus freezing, implementation mechanics for the three RAG scenarios, telemetry tracking, and the blind tripartite judgment protocol. Chapter 4 presents empirical results and Bayesian analysis, detailing computational latency decomposition, token consumption scaling, packing dynamics, retrieval efficacy, and semantic accuracy modeling. Finally, Chapter 5 formally validates the experimental hypotheses, discusses software engineering implications, addresses threats to validity, and outlines directions for future research.

THEORETICAL FOUNDATIONS AND RELATED WORK

This chapter establishes the conceptual and formal framework required to understand the experimental design and interpret its empirical results. The discussion is structured around the operational principles of Large Language Models and the execution physics of self-attention, extending to the formalization of the Retrieval-Augmented Generation (RAG) paradigm. Next, we address document layout analysis strategies, graph-based approximate nearest neighbor vector indexing, and the cognitive and attentional constraints associated with multi-hop reasoning over long contexts. The chapter concludes with the theoretical foundations of multi-evaluator semantic judging panels and the Bayesian inferential framework employed for statistical hypothesis testing.

2.1 AUTOREGRESSIVE NEURAL LANGUAGE MODELS AND ATTENTION DYNAMICS

The Transformer architecture (Vaswani *et al.*, 2017) relies fundamentally on attention mechanisms to map statistical dependencies across token sequences without recurrence or convolutional operators. In autoregressive language modeling, the probability distribution over the next sequence element is strictly conditioned on antecedent tokens.

Under Scaled Dot-Product Attention, an input sequence of length N is linearly projected into query matrices $Q \in \mathbb{R}^{N \times d_k}$, key matrices $K \in \mathbb{R}^{N \times d_k}$, and value matrices $V \in \mathbb{R}^{N \times d_v}$, where d_k represents the projection subspace dimensionality. The attention affinity matrix and the output representation are obtained analytically as:

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

where the dot product QK^T yields an $N \times N$ pairwise compatibility matrix, imposing quadratic computational and spatial asymptotic complexity $\mathcal{O}(N^2)$ relative to the processed context length. Multi-Head Attention extends this formulation by projecting representations across h independent linear subspaces, enabling the model to jointly attend to information across heterogeneous latent spaces (Vaswani *et al.*, 2017).

At the hardware execution level, LLM inference decomposes into two distinct

physical phases characterized by fundamentally different computational regimes (Dao *et al.*, 2022; Kwon *et al.*, 2023; Pope *et al.*, 2023):

1. **Prompt prefill phase:** all input context tokens are ingested and computed in parallel on the GPU. This stage is dominated by General Matrix Multiply (GEMM) tensor operations and is strictly compute-bound on tensor cores. During this phase, key and value activation tensors are populated and allocated into the Key-Value (KV) cache;
2. **Autoregressive decoding phase:** output tokens are generated sequentially, one token per forward pass. In each generative step, only the query projection for the newly emitted token is computed, while existing keys and values are read from the KV cache. Generation throughput is strictly memory-bandwidth bound, as every emitted token requires loading all model parameters and cached activations from Video RAM (VRAM) into compute registers (Shazeer, 2019; Patel *et al.*, 2024).

This physical asymmetry explains the latency profile in generative pipelines: variations in retrieved context length impact the GPU runtime almost entirely during the prefill phase, while the incremental decoding cost per generated token depends solely on the output answer length.

2.2 THE RAG PARADIGM AND NON-PARAMETRIC MEMORY MODELING

Knowledge retention in Large Language Models is traditionally confined to synaptic weights adjusted during pre-training, representing static parametric memory (Petroni *et al.*, 2019; Roberts; Raffel; Shazeer, 2020). While state-of-the-art models encode vast factual corpora, exclusive reliance on parametric representations introduces severe software engineering failure modes:

1. **Factual hallucinations:** outputting syntactically fluent statements that lack empirical support or contradict grounded facts (Ji *et al.*, 2023; Huang *et al.*, 2023);
2. **Static temporal obsolescence:** structural inability to access proprietary data, private corpora, or domain updates post-dating training cutoff dates (Lazaridou *et al.*, 2021);
3. **Epistemological opacity and absence of audit trails:** diffuse parametric weight representations prevent deterministic verification of which specific source documents substantiate a given claim (Ribeiro; Singh; Guestrin, 2016; Doshi-Velez; Kim, 2017).

To overcome these constraints without resorting to continuous fine-tuning, Lewis et al. (Lewis *et al.*, 2020) introduced Retrieval-Augmented Generation (RAG), coupling a dense retrieval module with an autoregressive sequence-to-sequence generator conditioned on dynamic, external non-parametric memory.

Formally, given an input query x , the retriever queries an indexed document corpus $\mathcal{D}$ to fetch the top- k relevant passages $z \in \text{top-}k(P(z|x))$. The conditional probability of the output sequence $y = (y_1, \dots, y_T)$ under the token-level formulation (RAG-Token) is obtained by marginalizing over the k retrieved passages:

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

In this expression, $P(z|x)$ denotes the retrieval probability in the dense embedding space, while $P(y_t|x, z, y_{1:t-1})$ represents the generative probability conditioned jointly on the input query, preceding generated tokens, and external context passages z .

Modern taxonomies categorize RAG architectures into three evolutionary generations (Gao *et al.*, 2023; Yu *et al.*, 2024):

1. *Naive RAG*: a linear, static pipeline comprising fixed-size text chunking, dense vector similarity search, and direct concatenation of retrieved chunks into the prompt without structural transformation (Barnett; Bikdash; Kapoor, *et al.*, 2024);
2. *Advanced RAG*: incorporates specialized pre-retrieval optimizations (such as query rewriting and expansion) and post-retrieval processing (such as cross-encoder re-ranking and context compression) (Ma *et al.*, 2023; Xu; Shi; Choi, 2024);
3. *Modular RAG*: highly configurable architectures featuring dynamic routing, multi-source orchestration, and decoupled indexing versus generation granularities via hierarchical retrieval.

Selecting between these paradigms involves direct software engineering tradeoffs among factual accuracy, end-to-end response latency, and GPU VRAM residency.

2.3 DOCUMENT LAYOUT ANALYSIS AND TEXT STRUCTURAL ENGINEERING

Text chunking determines the granularity of information units submitted to both vector indexing and generative language models. Traditional linear chunking, which splits raw text into fixed character or token counts using a sliding window with overlap, introduces three severe structural failure modes (Auer *et al.*, 2024):

1. Syntactic and coreference fragmentation: arbitrary boundary cuts bisect sentences, dismembering grammatical structures and severing anaphoric references necessary for semantic comprehension;
2. Destruction of tabular and relational geometry: blind linear splitting destroys row and column relationships, rendering structured tables into disjointed and confusing token streams;
3. Loss of hierarchical discourse context: stripping section headers and parent titles detaches retrieved paragraphs from their original topical scope within the source document.

To resolve these defects, Document Layout Analysis (DLA) leverages computer vision models and structural parsers to identify functional layout primitives across complex pages, including headers, paragraphs, lists, and tables (Zhong; Tang; Yepes, 2019; Wang *et al.*, 2022). Modern document converters, such as Docling (Auer *et al.*, 2024), transform unstructured formats into semantically rich hierarchical Markdown, preserving native paragraph boundaries and tabular structures.

This structural fidelity enables two-tier Parent-Child Retrieval architectures, resolving the classical tradeoff between retrieval specificity and context completeness:

- **Child nodes (granular chunks):** compact fragments (e.g., 128 tokens) optimized for maximal semantic density and pinpoint accuracy during approximate nearest neighbor vector search;
- **Parent nodes (complete sections):** coherent document sections linked relationally to child nodes, fetched deterministically from a key-value document store at query time to supply the generator with broad, unbroken context.

By fetching complete parent sections alongside automatic metadata-based section deduplication, two-tier architectures mitigate contextual redundancy and prevent arbitrary prompt cuts caused by token window limits.

2.4 DENSE VECTOR INDEXING AND APPROXIMATE NEAREST NEIGHBOR SEARCH

Dense retrieval maps textual passages into fixed-dimensional continuous embedding spaces $\mathbf{e} \in \mathbb{R}^d$, placing semantically similar passages in close proximity within a metric latent space (Reimers; Gurevych, 2019; Muennighoff *et al.*, 2023).

In contrast to traditional lexical term-matching algorithms such as BM25 (Robertson; Zaragoza, 2009), dual-encoder dense embedding models, such as nomic-embed-text (Nussbaum *et al.*, 2024), are trained using contrastive objective functions to capture latent semantic relationships beyond strict lexical overlap. The metric similarity between two embedding vectors $\mathbf{u}$ and $\mathbf{v}$ is commonly evaluated via Cosine Similarity:

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

When embedding vectors are unit-normalized ($\|\mathbf{u}\|_2 = \|\mathbf{v}\|_2 = 1$), angular similarity reduces to the dot product $\mathbf{u} \cdot \mathbf{v}$, and squared Euclidean distance relates directly via $\|\mathbf{u} - \mathbf{v}\|_2^2 = 2(1 - \mathbf{u} \cdot \mathbf{v})$. Vector retrieval thus simplifies geometrically to dense inner-product computation.

In large-scale collections comprising thousands of passages, exhaustive brute-force search (k -NN) incurs linear time complexity $\mathcal{O}(N \cdot d)$, rendering real-time interactive search computationally prohibitive. Modern vector database engines, such as ChromaDB, utilize Approximate Nearest Neighbor (ANN) search algorithms implemented on optimized graph libraries like `hnswlib` (Pan; Wang; Li, 2024; Han; Liu; Wang, 2023).

Among these structures, the Hierarchical Navigable Small World (HNSW) graph (Malkov; Yashunin, 2020) represents the state-of-the-art. HNSW constructs a multi-layer hierarchy of proximity graphs: upper layers contain sparse long-range edges for rapid metric space traversal, while lower layers feature dense local connections for fine-grained convergence. Search traversals descend hierarchically with sublinear logarithmic time complexity $\mathcal{O}(\log N)$. This asymptotic behavior explains why dense vector search operates in milliseconds and does not represent the execution bottleneck in local RAG pipelines.

2.5 MULTI-HOP REASONING AND CONTEXTUAL DEGRADATION (LOST IN THE MIDDLE)

Single-hop question answering requires retrieving an isolated factual statement from a single passage. Conversely, multi-hop question answering requires systems to discover, link, and synthesize complementary evidence spread across multiple independent documents (Welbl; Stenetorp; Riedel, 2018; Yang *et al.*, 2018). Conceptually, multi-hop tasks model latent reasoning graphs in which a retrieval failure on any single link disrupts the entire inferential chain.

These multi-hop queries manifest primarily across two distinct cognitive structures:

1. **Bridge questions:** where identifying the target entity depends on discovering and

resolving an intermediate relational entity introduced in a separate document;

2. **Comparison questions:** requiring the simultaneous extraction and comparative evaluation of properties across two or more distinct entities under temporal, numerical, or qualitative criteria.

The HotpotQA benchmark (Yang *et al.*, 2018) serves as the canonical standard for evaluating multi-hop reasoning. In its official distractor setting, each query is paired with ten Wikipedia articles: two official gold supporting articles and eight confounding distractor articles specifically chosen to stress-test retrievers against spurious lexical and semantic overlaps. To ensure internal validity, historical benchmarks must prevent Wikipedia editorial drift (Yao *et al.*, 2023) by freezing the corpus at the benchmark’s original reference cutoff (October 1, 2017), preventing historical contradictions caused by post-hoc edits.

When aggregating multiple retrieved passages into the prompt to fulfill multi-hop information needs, language models encounter the attentional bias known as the Lost-in-the-Middle effect (Liu *et al.*, 2024). This operational defect arises from the non-uniform distribution of self-attention weights across long context sequences: autoregressive transformers preferentially attend to tokens placed at the beginning (primacy bias) and end (recency bias) of the prompt, yielding a characteristic U-shaped retrieval performance curve. As redundant passages and retrieval noise accumulate in the central region of the context window, attention dispersion causes the model to overlook critical evidence, significantly degrading generation fidelity (Xiao *et al.*, 2024).

2.6 TRIANGULATED SEMANTIC EVALUATION AND BAYESIAN STATISTICAL INFERENCE

Performance assessment in NLP has historically relied on deterministic lexical overlap metrics, such as Exact Match (EM) and unigram F_1 -score (Papineni *et al.*, 2002; Lin, 2004). However, lexical metrics exhibit severe limitations when applied to modern generative models (Callison-Burch; Osborne; Koehn, 2006; Novikova *et al.*, 2017): fully correct answers that utilize valid paraphrases, synonyms, or descriptive sentences are unjustly penalized if they diverge from the static gold reference string.

To overcome lexical rigidity without incurring the latency and expense of continuous human annotation, Zheng *et al.* (Zheng *et al.*, 2023) formalized the LLM-as-a-Judge paradigm, employing frontier language models as automated semantic evaluators. To safeguard internal validity and prevent algorithmic biases, evaluation protocols must enforce three methodological safeguards (Gu *et al.*, 2024):

1. **Procedural double-blind masking:** complete anonymization of the generating architecture and global shuffling of instances to prevent positional and model-family favoritism;
2. **Structured few-shot chain-of-thought rationales:** mandatory generation of step-by-step reasoning verified against official gold supporting facts prior to outputting numerical scores;
3. **Multi-evaluator blind triangulation:** deployment of a tripartite panel composed of a human reference auditor and two independent synthetic judges from competing providers, eliminating single-model self-enhancement bias.

Inter-rater reliability across the panel is quantified using the Intraclass Correlation Coefficient (ICC), a rigorous measurement theory metric (Shrout; Fleiss, 1979; Koo; Li, 2016). For designs where a fixed panel of k judges evaluates all instances under an absolute agreement criterion for the mean rating, the two-way mixed-effects average-measures model ($ICC(3, k)$) is computed via two-way ANOVA as:

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

where MS_R denotes mean square between evaluated targets and MS_E represents residual error mean square. Values exceeding 0.75 denote excellent inter-rater reliability, demonstrating statistical measurement robustness (Koo; Li, 2016).

In statistical hypothesis testing, classical Null Hypothesis Significance Testing (NHST) based on p -values exhibits well-documented limitations in empirical Software Engineering, including artificial sensitivity to sample size and inability to quantify effect magnitude under operational uncertainty (Wasserstein; Lazar, 2016; Kitchenham *et al.*, 2017).

Consequently, this research adopts Bayesian estimation with full posterior distributions, coupling 95% Highest Density Intervals (HDI 95%) with declared Regions of Practical Equivalence (ROPE) (Kruschke, 2018; McElreath, 2020). Under this framework, pairwise differences between RAG architectures are formally resolved by evaluating posterior density relative to the ROPE bounds:

1. Substantial practical difference is declared when the 95% HDI falls entirely outside the $[-ROPE, +ROPE]$ interval;
2. Strict practical equivalence is accepted when the 95% HDI lies entirely within the ROPE interval;

3. Indecision or sampling uncertainty is recorded when the interval overlaps the ROPE boundary or the null value, with the direct posterior probability of superiority $P(\theta > 0 \mid \mathcal{D})$ reported as a continuous decision support metric.

EXPERIMENTAL METHODOLOGY

This chapter details the experimental methodology, software architecture, and control protocols implemented in the benchmark. We describe the modular automation pipeline, the stratified sampling and historical corpus freezing protocols, the implementation details of the three evaluated RAG architectures, hardware telemetry monitoring, and the double-blind tripartite semantic evaluation panel.

3.1 EXPERIMENTAL ARCHITECTURE AND EVALUATED RAG SCENARIOS

3.1.1 Overview and Modular Software Architecture

The experimental platform was developed in Python (version 3.11.9) under a modular architecture strictly adhering to the separation of concerns, designed to ensure strict scientific reproducibility, execution isolation, and unconfounded variable control.

The pipeline executes through a deterministic suite of ten sequential scripts (**s01** through **s10**), organized across four independent methodological phases:

1. **Corpus preparation (s01 to s04):** encompasses semantic filtering of the *HotpotQA* dataset, complexity-stratified sampling ($N = 180$), automated scraping of corresponding historical Wikipedia articles, and structural conversion via Docling;
2. **Vector indexing and docstore population (s05):** builds isolated collections in ChromaDB for each experimental group and persists the relational parent-mapping docstore to disk;
3. **Paired execution and telemetry (s06 and s07):** executes the calibration pilot and the main battery of serialized inferences under unitary concurrency ($N = 1$), continuously logging physical GPU thermal telemetry and phase-decomposed latencies;
4. **Analytical consolidation and auditing (s08 to s10):** aggregates raw JSON logs into structured relational tables, orchestrates the double-blind tripartite LLM-as-a-Judge evaluation protocol, and conducts programmatic data integrity verification.

To avoid algorithmic redundancy and enforce cross-stage metric consistency, the architecture is underpinned by shared infrastructure modules and a central inference engine:

- `common.py`: centralizes configuration loading from `config.yaml`, audit logging, random seed freezing (`seed = 42`), and SQLite database management (`corpus.db`);
- `metrics.py`: standardizes text normalization routines and computes deterministic retrieval and generation metrics (*Recall@k*, *Hit Rate*, *Exact Match*, *Containment Match*, and lexical F_1 -score);
- `engine.py`: core inference orchestrator that interfaces with the local Ollama daemon (`llama3.1:8b-instruct-q4_K_M`), switches between RAG retrieval mechanisms, and enforces dynamic token budget constraints.

Figure 3.1 illustrates the physical directory structure and modular code organization of the project:

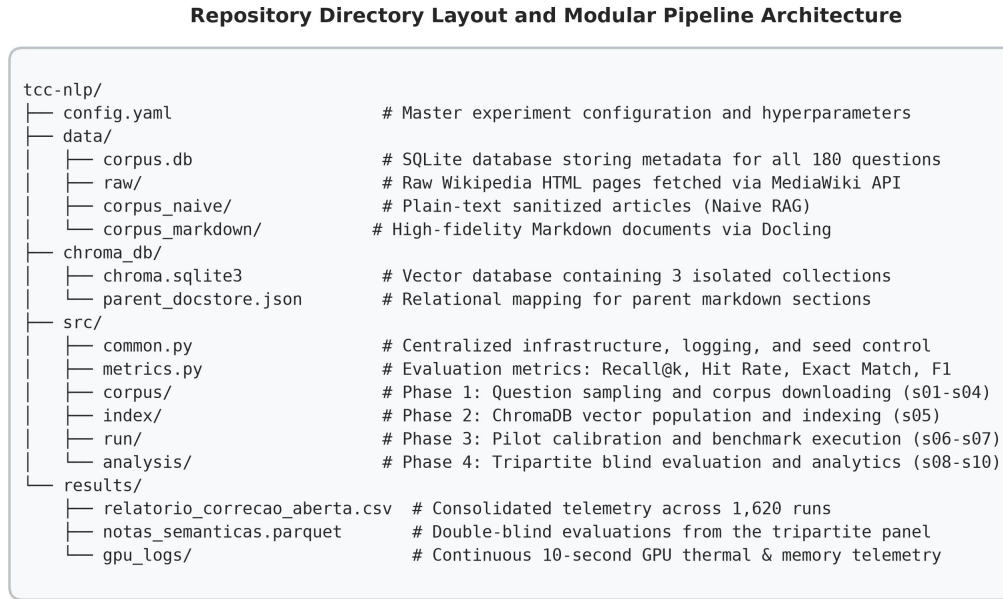


Figure 3.1: Repository directory layout and modular pipeline architecture.

Source: Authors (2026).

3.1.2 End-to-End Data Flow

The experimental pipeline enforces a unidirectional, decoupled, and fully auditable data flow designed to ensure strict artifact traceability. Each computational phase consumes

validated inputs and emits immutable intermediate disk artifacts before proceeding to subsequent stages, eliminating state corruption in volatile memory. As illustrated in Figure 3.2, the data flow unfolds across four operational phases:

1. **Corpus acquisition and structuring:** processes raw *HotpotQA* records, executes difficulty-stratified sampling, scrapes historical Wikipedia articles, converts them to structured Markdown via Docling, and persists supporting tables in SQLite (`corpus.db`);
2. **Vector indexing and persistence:** computes dense embeddings via *nomic-embed-text*, populates the three ChromaDB collections independently, and serializes the parent docstore mapping to disk;
3. **Paired inference and hardware telemetry:** submits queries to the local Llama 3.1 8B model via Ollama under serialized execution ($N = 1$), randomly permuting scenario order per query to prevent cache and thermal bias while logging hardware telemetry;
4. **Metric consolidation and evaluative triangulation:** consolidates raw execution logs into master tables, submits generated answers to the double-blind tripartite panel (human reference auditor, GPT-5, and Gemini 3.7 Pro), and computes psychometric agreement metrics via *ICC*(3,3).

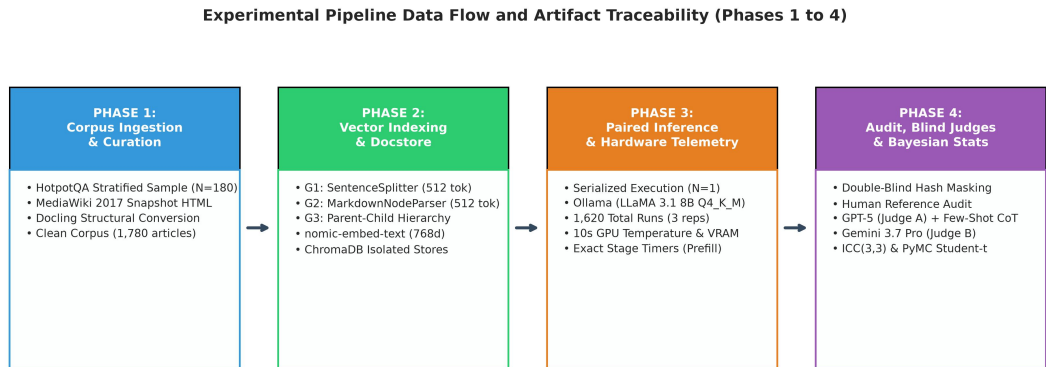


Figure 3.2: Experimental pipeline data flow and intermediate artifact traceability.

Source: Authors (2026).

3.1.3 Characterization of Evaluated RAG Scenarios and Ablation Design

To isolate the impact of individual data engineering strategies, the benchmark structures interventions across three paired treatment groups. Vector embeddings were indexed into separate, isolated ChromaDB collections (`col_naive`, `col_structured`, and

`col_hierarchical`), preventing cross-contamination during search. The comparative partitioning, indexing, and retrieval workflow is illustrated in Figure 3.3:

1. **Group 1 (Naive RAG, Control):** establishes the canonical baseline. It operates over plain-text files (`.txt`) partitioned using a linear sliding window via `SentenceSplitter`, fixed at 512 tokens with a static overlap of 50 tokens. Vector search is conducted via cosine similarity directly over these linear chunks;
2. **Group 2 (Structured RAG):** isolates the effect of document layout conversion and section-bounded segmentation. Articles are converted to hierarchical Markdown via `Docling`, preserving headers (`#`, `##`, `###`), lists, and tables. Partitioning via `MarkdownNodeParser` respects native section boundaries, applying secondary linear splitting only if an individual section exceeds the 512-token ceiling. Vector search operates identically to G1;
3. **Group 3 (Hierarchical RAG):** evaluates the incremental benefit of two-tier parent-child retrieval on structured data. It adopts the complete sections from G2 as parent nodes (up to 512 tokens), from which smaller 128-token child nodes are derived using `SentenceSplitter` (20-token overlap). Only child nodes are indexed in ChromaDB. At query time, retrieved child nodes are deterministically replaced by their corresponding complete parent sections stored in `parent_docstore.json`, with automatic identifier deduplication to eliminate redundancy.

This study establishes an incomplete factorial design governed by planned linear contrasts, enabling causal decomposition of experimental effects:

- **Structural contrast ($G2 - G1$):** holding direct vector search constant, measures the net effect of Markdown layout preservation and section-bounded chunking;
- **Retrieval contrast ($G3 - G2$):** holding the structured corpus constant, isolates the additive effect of granular child indexing (128 tokens) coupled with deterministic parent section expansion.

By contrast additivity, the global benchmark effect ($G3 - G1$) decomposes into $(G2 - G1) + (G3 - G2)$, attributing differences in latency, token consumption, and generative fidelity to specific architectural stages. Internal validity is maintained via two strict experimental controls:

1. **Parity of retrieval depth and packing policy:** retrieval depth is strictly fixed at $k = 20$ candidates across all three groups. Prompt injection follows a greedy packing policy, filling available context up to the 8,192-token ceiling of Llama 3.1 8B;

2. **Textual identity of candidate passages:** parent sections retrieved in Group 3 are strictly identical to the document blocks comprising Group 2, ensuring that the $G3 - G2$ contrast evaluates topological navigation in embedding space without confounding variations in underlying text.

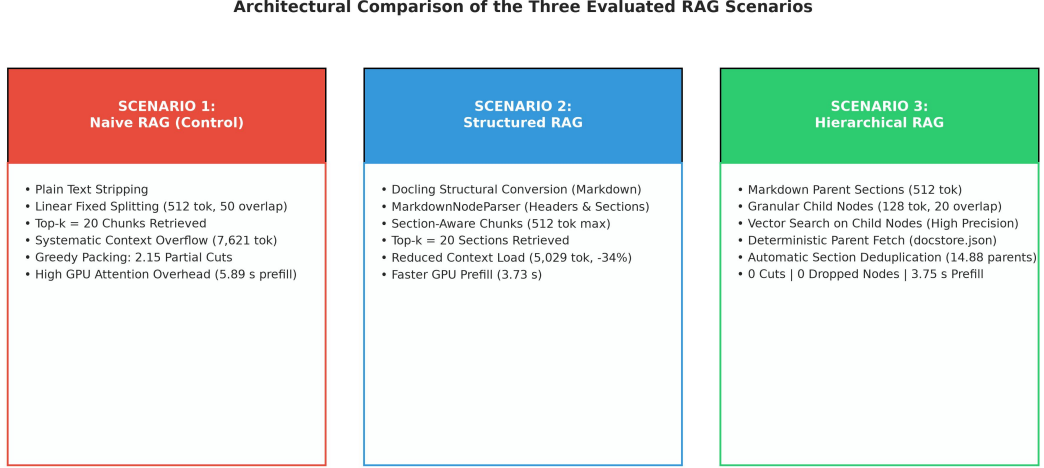


Figure 3.3: Architectural comparison of the three evaluated RAG scenarios.

Source: Authors (2026).

3.1.4 Hierarchical Parent Section Resolution

In Group 3 (Hierarchical RAG), retrieval utilizes a deterministic parent section resolution protocol. Unlike standard LlamaIndex `AutoMergingRetriever` (which requires a majority of child nodes from a section to be retrieved before triggering parent promotion), our implementation immediately resolves every retrieved top- k child node to its complete parent section in `parent_docstore.json`. This methodological choice ensures experimental isolation, guaranteeing that any relevant child node provides complete narrative context to the generator.

To avoid redundant context injection and wasted token budget, the pipeline enforces strict deduplication: multiple child nodes belonging to the same section result in a single parent section being included in the prompt. For ranking, the parent section inherits the cosine similarity score of its highest-ranked child node.

3.1.5 Context Window Management and Dynamic Token Budgeting

Prompt construction across all three groups follows a dynamic token budgeting algorithm, replacing character-based heuristics. The remaining capacity available for

retrieved nodes ($C_{\text{available}}$) is computed deterministically as:

$$C_{\text{available}} = C_{\text{total}} - T_{\text{system}} - T_{\text{query}} - T_{\text{predict}} - M_{\text{margin}},$$

where:

- $C_{\text{total}} = 8,192$ tokens is the context window allocated in Ollama (`num_ctx`) for Llama 3.1 8B;
- $T_{\text{predict}} = 512$ tokens is the maximum generation cap (`num_predict`);
- $M_{\text{margin}} = 64$ tokens absorbs prompt formatting overhead and special tokens (`margem_contexto_tokens`);
- T_{system} and T_{query} denote token lengths of the system prompt and evaluated query.

Parameters T_{system} and T_{query} are pre-estimated in microseconds via Python’s `tiktoken` library (`cl100k_base` encoding). Because Ollama operates as a decoupled server without a synchronous partial tokenization endpoint during prompt construction, `tiktoken` serves as a lightweight proxy to guide node packing and prevent context overflow. The 64-token safety margin absorbs vocabulary discrepancies between `cl100k_base` (100k tokens) and Llama 3.1’s native BPE tokenizer (128k tokens). Crucially, this approximation applies solely during prompt assembly: all official, audited token counts reported in this study derive from native Ollama GPU telemetry counters (`prompt_eval_count` and `eval_count`).

Retrieved fragments are allocated into the prompt in descending order of similarity under a greedy packing policy. When a node exceeds remaining budget, it is proportionally truncated to fit residual capacity (`nodes_partial`). If this conservative cut leaves remaining token budget, the algorithm continues evaluating subsequent nodes, slicing additional fragments until the context window is exhausted. Nodes for which zero budget remains are dropped entirely (`nodes_dropped`). This policy explains the average of 2.15 partial nodes observed in *Naive RAG*, preventing VRAM overflow and eliminating out-of-context execution failures.

3.2 STRATIFIED SAMPLING AND QUERY SELECTION

3.2.1 Data Source and Domain Scope

The evaluation benchmark was sampled from the official *HotpotQA* dataset (Yang *et al.*, 2018) in its distractor setting (train and validation splits). HotpotQA was selected

because it mandates multi-hop reasoning (requiring evidence synthesis across multiple documents), a regime where retrieval efficacy directly dictates generative model performance.

To ensure strict reproducibility, data loading pins the Hugging Face commit hash to 1908d6afbbeat072334abe2965f91bd2709910ab.

The experimental domain is centered on encyclopedic Wikipedia topics (history, geography, science, biographies, and arts), satisfying two methodological criteria:

1. **Semantic diversity:** subjects dense retrieval to diverse named entities and relational structures, mirroring realistic production workloads;
2. **Continuous narrative prose:** prioritizes structured prose with sections, lists, and tables, avoiding domain-specific notations (e.g., source code or advanced mathematical formulas) that would necessitate specialized parsers.

3.2.2 Difficulty Stratification and Deterministic Sampling

Sampling was performed across official HotpotQA partitions to guarantee balanced cognitive complexity. The final benchmark comprises $N = 180$ questions, equally distributed across native difficulty strata:

- 60 Easy questions;
- 60 Medium questions;
- 60 Hard questions.

Sampling was conducted via `src/corpus/s02_sample_stratified.py` using canonical identifier sorting and a fixed seed (`seed = 42`). The algorithm preserves the dataset’s native balance between bridge and comparison reasoning types.

3.2.3 Human Validation and 1:1 Replacement Protocol

Following algorithmic sampling, the candidate pool underwent pre-experimental human inspection (`data/questions/validacao_manual.csv`) against three soundness criteria:

- **Factual verifiability:** the question admits an unambiguous, verifiable ground truth answer;
- **Ground truth consistency:** the reference answer faithfully reflects source text and historical consensus;

- **Evidential completeness:** annotated supporting facts are sufficient to sustain the multi-hop deductive chain.

During initial pilot screening (20 Easy, 20 Medium, 20 Hard), ten questions presenting irreconcilable ambiguities were replaced with the immediately subsequent record from the same difficulty stratum, maintaining strict 1:1 replacement and deterministic sampling. The expanded sample ($N = 180$) was programmatically audited for supporting fact integrity, and all 540 final generated answers were manually reviewed.

3.3 CORPUS CONSTRUCTION, SANITIZATION, AND INTEGRITY AUDITING

3.3.1 Historical Freezing and Data Drift Mitigation

To build the closed knowledge corpus, all article titles present in the HotpotQA context fields (supporting facts and distractors) were extracted.

Direct scraping of modern Wikipedia would introduce severe editorial drift (Yao *et al.*, 2023). HotpotQA was constructed from an official Wikipedia dump dated October 1, 2017. Preliminary tests against modern Wikipedia revealed that 75.3% (113 of 150) of gold supporting sentences could no longer be found as exact textual matches due to years of collaborative edits.

To eliminate this bias, a historical freezing protocol was implemented in `src/corpus/s03_download_articles.py`:

- Scrapes the MediaWiki Action API (`action=query&prop=revisions`) in reverse chronological order (`rvdir=older`) from timestamp `rvstart=2017-10-01T00:00:00Z`;
- Identifies the revision ID (`oldid`) and timestamp of the last version published prior to the cutoff date;
- Fetches the complete HTML for that exact historical revision and freezes it locally in `data/raw/`, recording metadata in `corpus.db`;
- **Download success rate and error audit:** of 1,788 unique titles required for the corpus (358 gold supporting articles and 1,430 distractors), 1,780 were successfully downloaded. The 8 failed downloads (`download_report.json`) were due to deleted or moved pages lacking pre-2017 revisions. Database auditing in `corpus.db` confirmed that 100% of the 8 missing articles were non-essential distractors, with zero loss among gold supporting articles.

3.3.2 DOM Pre-Sanitization and Format Parity

Before generating files for the three groups, scraped HTML undergoes sanitization to remove artifacts that could distort vector embeddings:

- Strips phonetic guides and inline metadata (`<span class="noexcerpt">`, IPA pronunciations) and footer navigation boxes (`.navbox`);
- Strips footnote superscripts (`<sup>`) and unwraps anchor tags (`<a>`), preserving plain text without `href` attributes;
- Injects a synthetic `<h2>Introduction</h2>` tag at the start of `<body>` to prevent Docling from discarding lead paragraphs prior to the first section heading.

From this sanitized HTML, the pipeline produces two parallel formats: normalized plain-text `.txt` files (G1) with stripped tables and tags, and structured `.md` files (G2 and G3) preserving headings, lists, and tables.

3.3.3 Automated Corpus Audit and Empirical Coverage Bounds

Corpus integrity was validated via automated auditing in `src/corpus/s04_prepare_corpus.py` (`results/audit/corpus_audit.json`):

- Exact normalized substring search (Unicode NFKC, casefolding, collapsed punctuation) located 98.14% (421 of 429) of gold supporting facts within generated files;
- Manual review of the 8 residual facts confirmed that all 8 were present in the downloaded text, with automated matching failing only due to minor whitespace and abbreviation differences inherited from the original dataset extractor.

Auditing confirmed that fact representation affected plain-text (G1) and structured (G2/G3) formats identically, ensuring high internal validity.

3.4 OPERATIONAL ENVIRONMENT, HARDWARE, AND SOFTWARE INFRASTRUCTURE

3.4.1 Hardware Specifications and Experimental Isolation

To ensure strict reproducibility and isolation against network and operating system contention, all benchmark phases (1 through 4) executed on a dedicated local workstation under unitary concurrency ($N = 1$) with zero background processes:

- **CPU:** Intel Core i7-10700 (8 physical cores / 16 threads, 2.90 GHz base, Turbo Boost to 4.80 GHz, 16 MB Smart Cache, x86_64);
- **GPU:** NVIDIA GeForce RTX 3050 (8 GB dedicated GDDR6 VRAM, 2,560 CUDA cores, driver v596.36), maintaining steady ~ 4.7 GB VRAM residency during Llama 3.1 8B inference;
- **RAM:** 32 GB DDR4 (operating frequency 2,933 MHz);
- **Storage:** Western Digital Blue 1 TB HDD (WD10SPZX-21Z10T0, SATA III 6 Gb/s, 5,400 RPM, 128 MB cache buffer);
- **OS:** Microsoft Windows 11 Home (64-bit, build 26200);
- **Runtime:** Python v3.11.9 in an isolated virtual environment (`.venv`) and local Ollama inference server (v0.3+) with CUDA acceleration.

Table 3.1 summarizes the software ecosystem, versions, and methodological roles across the benchmark pipeline.

3.4.2 Language Models, Embeddings, and Parameter Configuration

The architecture was parameterized to decouple local inference execution from cloud-based evaluation:

- **Local generator model (llama3.1:8b):**
 - *Architecture:* LLaMA 3.1 8B Instruct (Meta AI), 4-bit quantized (Q4_K_M) via local Ollama with CUDA;
 - *Temperature:* set strictly to 0.0 (deterministic greedy decoding);
 - *Context window (`num_ctx`):* 8,192 tokens during RAG inference;
 - *Generation ceiling (`num_predict`):* maximum 512 tokens;
 - *VRAM persistence:* `keep_alive`: -1, locking model weights into VRAM (~ 4.7 GB) to prevent reload latency spikes.
- **Dense embedding model (nomic-embed-text):**
 - *Architecture:* Nomic Embed Text v1.5 (Nomic AI), local Ollama instance;
 - *Context window:* 4,096 tokens (`embedding_num_ctx`: 4096), accommodating nearly all sections (only 35 outlier sections truncated: 22 in Structured, 13 in Hierarchical);
 - *Dimensionality and metric:* 768-dimensional dense vectors indexed with Cosine Similarity in ChromaDB.

Table 3.1: Software ecosystem, libraries, and experimental architecture components.

Component	Technology	Version / Config	Methodological Role in Experiment
Base Language	Python	v3.11.9	Pipeline orchestration and analytical execution (<code>s01</code> to <code>s10</code>).
Inference Server	Ollama	Daemon v0.3+ / Client v0.33.2	Local GPU execution of LLaMA 3.1 8B (Q4_K_M) and <code>nomic-embed-text</code> .
Vector Database	ChromaDB	v0.6.3 (local persistence)	Dense embedding storage across three isolated collections; cosine search.
Layout Parser	Docling (IBM)	v2.123.1 (HTML backend, OCR off)	Sanitized HTML to Markdown conversion preserving document hierarchy.
RAG Framework	LlamaIndex	v0.12.10 (core)	Text chunking (<code>SentenceSplitter</code>), hierarchical node abstractions, and docstore.
Token Proxy	Tiktoken	v0.8.0 (<code>cl100k_base</code>)	Microsecond token estimation for greedy prompt packing and context bounds.
Relational Database	SQLite	<code>sqlite3</code> (<code>data/corpus.db</code>)	Structured storage of question metadata, articles, and audit logs.
Statistical Modeling	PyMC	PyMC v5.16+ and SciPy v1.13+	Bayesian Student- <i>t</i> regression via NUTS and direct Monte Carlo simulation.
Evaluation Panel	LLMs and Human	Human Auditor, GPT-5, Gemini 3.7	Double-blind tripartite semantic evaluation panel (LLM-as-a-Judge).

Source: Authors (2026).

3.5 PAIRED EXECUTION PROTOCOL, RANDOMIZATION, AND BIAS CONTROLS

3.5.1 Paired Experimental Design and Sample Size

The benchmark execution via `src/run/s07_benchmark.py` utilizes a repeated-measures randomized block design. The 180 questions were evaluated across all three RAG scenarios (G1, G2, G3) over 3 technical repetitions ($R = 3$):

$$N_{\text{total}} = 180 \text{ questions} \times 3 \text{ treatments} \times 3 \text{ repetitions} = 1,620 \text{ inference runs.}$$

This design yields 540 unique Question $\times$ Scenario pairs, allowing each question to serve as its own control to absorb question difficulty heterogeneity in statistical modeling.

3.5.2 Order Randomization and Interleaved Global Rounds

To neutralize sequence bias, GPU thermal throttling, and ChromaDB/Ollama caching artifacts, the execution order was structured via `montar_plano()` (`src/run/engine.py`) under a fixed seed (`seed = 42`):

- **Interleaved global rounds:** the 3 repetitions executed as sequential macro blocks (Rounds 1, 2, and 3), completing all 540 pairs before the next round commenced;
- **Intra-question scenario randomization:** within each round, question order was shuffled deterministically. For each question, the three RAG scenarios executed consecutively in a randomized order (e.g., Question A: $G2 \rightarrow G1 \rightarrow G3$; Question B: $G3 \rightarrow G2 \rightarrow G1$). This guarantees that treatments for a given question execute under identical physical conditions while neutralizing order bias.

3.5.3 Strict Serial Concurrency Regime ($N = 1$)

Execution operated in strict serialized concurrency ($N = 1$):

- A new inference trial begins only after the preceding trial completes, its raw JSON log is atomically saved to disk (`results/raw_json/`), and volatile buffers are cleared;
- Eliminates contention for PCIe bus bandwidth, CUDA cores, and VRAM, ensuring latency metrics reflect isolated algorithmic costs.

3.5.4 Warm-Up Protocol and Cache Cold-Start Elimination

Prior to logging official trials, the runner executes a 3-question warm-up using out-of-sample questions (`aquecer()` in `engine.py`):

- Forces Ollama to load Llama 3.1 8B and `nomic-embed-text` weights into VRAM and allocate execution buffers;
- Discards warm-up metrics entirely, eliminating cold-start latency spikes from benchmark data.

3.5.5 Continuous Background Infrastructure Telemetry

During all 1,620 runs, a dedicated asynchronous thread (`MonitorGPU`) logged hardware state every 10 seconds (`results/gpu_logs/gpu_<timestamp>.csv`):

- **GPU:** package temperature (°C), allocated VRAM (MB), and core utilization via `nvidia-smi`;
- **Host memory:** physical RAM (MB) and pagefile/SWAP usage (MB) via `psutil`;
- **Stability verification:** confirmed zero thermal throttling, stable ~ 4.7 GB VRAM residency, and zero disk swapping throughout execution.

3.6 PERFORMANCE INDICATORS, TELEMETRY METRICS, AND DATA MAPPING

3.6.1 Context Retrieval Efficacy Metrics

To measure retriever efficacy, recovery of official HotpotQA gold supporting facts (Yang *et al.*, 2018) was quantified:

- **Evidence Recall@ k** (`recall_at_k`): proportion of required supporting facts (S_i) present within retrieved context chunks (R_i):

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

- **Strict Hit Rate** (`hit_rate`): binary indicator denoting whether the retriever

recovered 100% of required supporting facts for the multi-hop question:

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

- **Evidence audit and false negative mitigation:** while automated evaluation uses a dual-matching routine (`fato_presente()` in `src/metrics.py`, combining normalized substring matching and lexical overlap $\geq 75\%$), evidence recovery was audited manually (`validacao_manual_evidencias` in `results/relatorio_correcao_aberta.csv`).

3.6.2 Secondary Lexical Generation Quality Metrics (Descriptive Analysis)

Evaluation against ground truth answers was computed in `src/metrics.py` with text normalization (lowercasing, punctuation, and article removal) for classical comparability:

- **Containment Match (`containment_match`):** binary flag checking if all ground truth tokens are present within generated answer tokens (`all(token in pred_tokens for token in gold_tokens)`), accommodating complete sentences containing the target entity;
- **Exact Match (`exact_match`, **EM**):** strict equality (1.0) between normalized predicted string and reference string;
- **Lexical F_1 -Score (`f1_score`):** harmonic mean of unigram precision (P_{lex}) and recall (R_{lex}) between generated and reference answers:

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

Methodological note on F_1 : Because lexical F_1 systematically penalizes valid explanatory sentences produced by autoregressive models, it was not used as the confirmatory dependent variable for H_2 , which was evaluated via Triangulated Semantic Accuracy (Section 3.7).

3.6.3 Latency Telemetry and Execution Phase Decomposition

Inference latency was tracked at millisecond resolution across four execution components:

- **Retrieval Time** (T_{rec} / `tempo_recuperacao_sec`): time spent embedding the query and executing cosine search in ChromaDB (including parent section resolution in Group 3);
- **Context Ingestion Time** (T_{prefill} / `tempo_prefill_sec`): GPU time processing and ingesting compiled prompt tokens into the KV cache (`prompt_eval_duration`);
- **Generation Time** (T_{gen} / `tempo_geracao_sec`): autoregressive token-by-token decoding time (`eval_duration`);
- **Total Latency** (T_{total} / `tempo_total_sec`): end-to-end wall-clock time from query submission to final token decoding.

3.6.4 Context Allocation and Computational Cost Metrics

To monitor context window usage across the 8,192-token capacity, the following metrics were tracked:

- **Official Input Tokens** (`tokens_prompt` / `prompt_eval_count`): canonical token count processed on the GPU, reported by native Ollama API telemetry;
- **Official Output Tokens** (`tokens_resposta` / `eval_count`): tokens generated by Llama 3.1 8B;
- **Dynamic Budget Node Accounting:**
 - `nodes_returned`: nodes returned from vector search ($k = 20$);
 - `nodes_used`: nodes packed into prompt (full or partial);
 - `nodes_partial`: nodes proportionally truncated to absorb residual budget ($C_{\text{available}}$);
 - `nodes_dropped`: retrieved nodes discarded due to context ceiling exhaustion.

3.6.5 Master Dataset Consolidation and Artifact Layout

Raw execution data across all 1,620 runs (consolidated into 540 unique pairs for semantic evaluation) were unified in `results/relatorio_correcao_aberta.csv`, indexed by `numero_execucao` ($1 \leq i \leq 540$) across five analytical dimensions:

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

2. **Resource and latency telemetry:** `tempo_recuperacao_sec`, `tempo_prefill_sec`, `tempo_geracao_sec`, `tempo_total_sec`, `tokens_prompt`, and `tokens_resposta`;
3. **Context budgeting:** `blocos_retornados`, `blocos_usados`, `blocos_parciais`, and `blocos_descartados`;
4. **Retrieval and generation performance:** `fatos_recuperados`, `recall_at_k`, `hit_rate`, `containment_match`, `exact_match`, `f1_score`, `pergunta`, `gabarito_resposta`, and `resposta_gerada`;
5. **Validation metadata:** `validacao_manual_resposta`, `validacao_manual_evidencias`, and human review notes.

3.7 DOUBLE-BLIND EVALUATION PROTOCOL AND SEMANTIC QUALITY VALIDATION

3.7.1 Overcoming Lexical Limitations via Semantic Judging and H_2 Formalization

String-matching metrics like Exact Match and F_1 disproportionately penalize answers that express correct facts through valid paraphrasing or full sentences (e.g., “*The director of the film is Christopher Nolan*” versus reference “*Christopher Nolan*”).

Consequently, Triangulated Semantic Accuracy (Mean Score ≥ 8.0) was established as the primary confirmatory dependent variable for hypotheses H_{2a} and H_{2b} (modeled via Bayesian hierarchical logistic regression), relegating lexical F_1 and Exact Match to secondary descriptive metrics.

3.7.2 Blinding Protocol and Anonymization

Evaluation followed a double-blind protocol in `src/analysis/s09_judge_protocol.py`:

- **Blind identifier anonymization (`blind_id`):** scenario names were replaced with opaque 12-character SHA-1 hashes;
- **Global shuffling:** instance order was globally randomized to prevent sequence and block bias;
- **Metadata omission:** evaluators were provided only with Question, Gold Supporting Sentences, Ground Truth, and Generated Answer;

- **Sampling for evaluation and decoding stability:** while latency analyses used medians across $R = 3$ repetitions, semantic evaluation utilized answers from the first technical run (`repeticao == 1`) across all 540 pairs. Due to seed freezing (`seed = 42`) and zero temperature ($T = 0.0$), textual reproducibility was 99.63% (538 of 540 pairs produced strictly identical character-by-character outputs across repetitions). In the two cases with minor floating-point divergence, the first trial output was preserved.

3.7.3 Tripartite Judge Triangulation Panel

The evaluation panel comprised a human reference auditor and two frontier LLMs from competing providers:

- **Human reference auditor:** manual audit by one of the authors under full blinding and global shuffling (`notas_humano.csv`);
- **Judge A (GPT-5, OpenAI):** algorithmic evaluator prompted with Few-Shot Chain-of-Thought, $T = 0.0$, and a standardized 0–10 rubric;
- **Judge B (Gemini 3.7 Pro, Google AI):** independent algorithmic judge from a competing provider to eliminate single-vendor evaluation bias.

AI judges operated with deterministic decoding ($T = 0.0$) and scored answers on a 0 to 10 scale with conceptual anchor rubrics (Few-Shot CoT), penalizing hallucinations and incomplete reasoning.

3.7.4 Inter-Rater Reliability and Methodological Safeguards

Inter-rater agreement was evaluated via the Intraclass Correlation Coefficient, $\text{ICC}(3, k)$ (two-way mixed-effects average-measures model) (Shrout; Fleiss, 1979):

$$\text{ICC}(3, k) = \frac{\text{MS}_R - \text{MS}_E}{\text{MS}_R},$$

where MS_R is the mean square between targets and MS_E is residual error mean square.

- **Validation threshold:** established *a priori* at $\text{ICC} \geq 0.75$, denoting excellent agreement;
- **Methodological safeguard:** if $\text{ICC}(3, k)$ fell below 0.75, LLM judgments would be disqualified, relying solely on human manual audit as ground truth.

3.7.5 Bayesian Inferential Modeling and Hypothesis Testing

Hypotheses were tested using paired Bayesian inference to accommodate data skewness and estimate posterior contrast distributions directly (McElreath, 2020):

- **Total Latency and Phases (H_3):** robust Student- t linear regression in *PyMC* (Abril-Pla *et al.*, 2023) using NUTS (4 chains, 2,000 draws, 1,000 warm-up, achieving $\hat{R} = 1.000$ and ESS > 5,000);
- **Prompt Token Consumption (H_4):** direct Monte Carlo sampling from a Student- t posterior with Jeffreys prior (20,000 draws; 95% intervals via 2.5% and 97.5% percentiles), permitted by deterministic token packing rules;
- **Retrieval Efficacy (H_1):** paired Bayesian model of Recall@ k differences across scenarios per question;
- **Binary Success Outcomes (H_1 and H_2):** Hit Rate and Triangulated Semantic Accuracy (Score ≥ 8.0) modeled via paired logistic regression with binomial likelihood, estimating difference in proportions (Δp), Odds Ratio (OR), and 95% HDI.

Confirmatory inference evaluated 95% Highest Density Intervals against declared Regions of Practical Equivalence (ROPE) (Kruschke, 2018):

1. **Latency ROPE:** set at $\pm 10\%$ of control mean (-0.840 s to $+0.840$ s), absorbing thermal and OS noise;
2. **Token ROPE:** set at ± 200 tokens ($\sim 2.6\%$ of *Naive RAG* mean);
3. **Decision criteria:** practical superiority required 100% of the 95% HDI to fall outside the ROPE in the hypothesized direction ($P = 100.0\%$). Practical equivalence required 100% of the 95% HDI to fall strictly inside the ROPE;
4. **Metrics without ROPE (Recall@ k and Semantic Accuracy):** evaluated via posterior difference (Δ), Odds Ratio, and probability of superiority $P(\Delta > 0)$. Credible intervals crossing zero indicate non-rejection of the null hypothesis. Primary contrasts evaluated structural conversion (G1 vs. G2) and hierarchical retrieval (G2 vs. G3), with G3 vs. G1 reported as derived estimates.

3.8 REPRODUCIBILITY, AUDITABILITY, AND ETHICAL CONSIDERATIONS

The benchmark architecture adheres to open science governance across three pillars:

1. **Parameter freezing:** all hyperparameters, random seeds (`seed = 42`), dependency versions, prompt templates, and local model weights (`llama3.1:8b-instruct-q4_K_M` and `nomic-embed-text`) are centralized in `config.yaml`;
2. **Audit trails and open data:** all pipeline stages produce persistent disk records in `results/`, including scraping logs (`download_report.json`), corpus verification (`corpus_audit.json`), judge reports (`judges_report.json`), GPU telemetry (`gpu_logs/`), and the master dataset (`relatorio_correcao_aberta.csv`);
3. **Ethical compliance and data sovereignty:** uses solely public, permissively licensed data (HotpotQA under CC BY-SA 4.0; Wikipedia under CC BY-SA 3.0), containing zero Personally Identifiable Information (PII). All vector retrieval and inference executed 100% locally on an isolated workstation, while external judge API calls transmitted only anonymized text stripped of experimental metadata.

3.8.1 Statement on the Use of AI Assistance in Code Implementation

Software engineering modules in this project (Docling ingestion, ChromaDB vector indexing, local Ollama orchestration, hardware telemetry logging, and PyMC statistical modeling) were implemented with the assistance of frontier language models (Claude and Gemini) through prompt engineering and architectural specifications provided by the authors.

All generated code was rigorously unit-tested, statically typed, and verified in local runs. Methodological conception, experimental design, hypothesis formulation, hardware control, and scientific interpretation remain the sole intellectual responsibility of the authors.

RESULTS AND DISCUSSION

4.1 COMPARATIVE ANALYSIS OF RESPONSE LATENCY AND HARDWARE EFFICIENCY

Total response latency constitutes one of the primary operational bottlenecks in deploying RAG architectures in production environments (Gao *et al.*, 2023). In this study, the latency of each inference was captured in isolation, under serialized concurrency ($N = 1$) with strict GPU thermal management and dedicated hardware telemetry, as detailed in Section 3.4.

The analysis was conducted on the median of paired repetitions across the 180 questions of the benchmark (540 consolidated trials), examining both the total inference time and its granular decomposition across the three canonical pipeline stages: (i) database vector retrieval, (ii) context ingestion and prompt evaluation in GPU attention memory (prefill), and (iii) autoregressive token decoding (generation), as summarized in Figure 4.1.

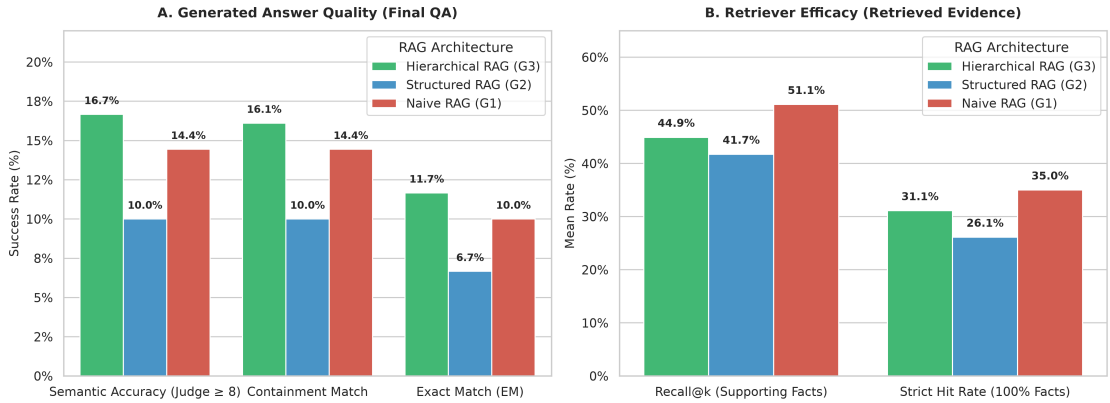


Figure 4.1: Comparative summary of overall performance across RAG scenarios.

Source: Authors (2026).

4.1.1 Global Latency Performance: Cross-Architectural Comparison

Descriptive results for total execution time reveal a substantial divergence between the control group (Naive RAG) and the approaches incorporating document pre-structuring (Structured RAG and Hierarchical RAG). Table 4.1 presents the descriptive statistics observed across the 180 questions of the corpus.

Table 4.1: Descriptive statistics of total inference latency across RAG architectures ($N = 180$).

RAG Architecture	Mean (s)	Standard Deviation (s)	Median (s)	Interquartile Range IQR (s)	Minimum (s)	Maximum (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, Control)	8.403	0.195	8.377	0.082	8.112	8.834

Source: Authors (2026).

As reported in Table 4.1, the control group (Naive RAG) yielded an average latency of 8.403 s, exhibiting substantially slower execution than the structured configurations. Furthermore, the Naive baseline showed remarkably low variance, with a standard deviation of only 0.195 s and an interquartile range of 0.082 s, a direct consequence of uniformly saturating the context window during the prefill phase. In contrast, *Hierarchical RAG* and *Structured RAG* achieved average execution times of 6.303 s and 6.278 s, respectively, providing a relative reduction of approximately 25.0% in overall response time, as illustrated by the empirical density profiles in Figure 4.2.

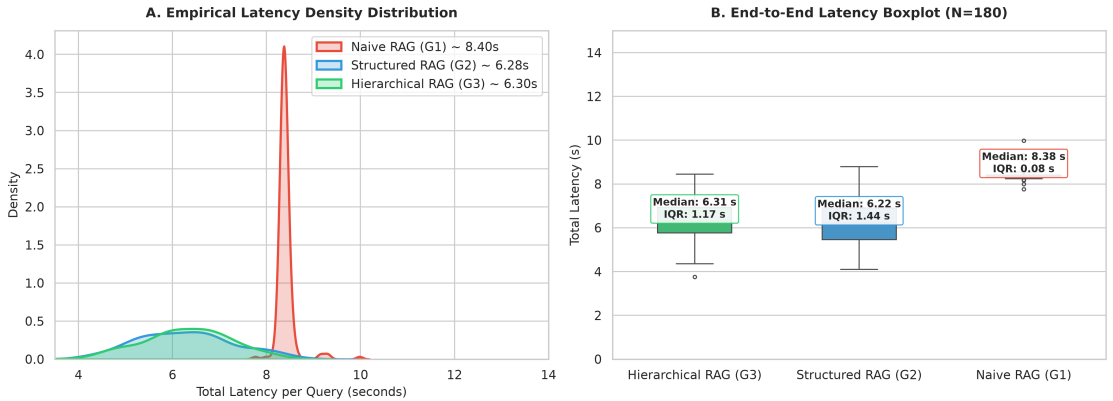


Figure 4.2: Empirical distribution of total response latency across RAG architectures.

Source: Authors (2026).

4.1.2 Bayesian Modeling of Paired Response Latency

To assess architectural effects within a rigorous probabilistic framework robust to distributional skewness and outliers, a paired Student- t regression was implemented via the *PyMC* library (Abril-Pla *et al.*, 2023; Kruschke, 2015). The model simultaneously estimated the posterior distributions of paired differences (Δ) across all three experimental configurations.

A Region of Practical Equivalence (ROPE) was defined at $\pm 10\%$ around the control group mean (Kruschke, 2018), spanning the interval $[-0.840 \text{ s}; +0.840 \text{ s}]$. Contrast differences whose 95% Highest Density Interval (HDI) falls entirely outside the ROPE provide evidence of practical engineering relevance rather than mere statistical significance. Table 4.2 summarizes the posterior parameters estimated across four MCMC chains (2,000 draws, 1,000 warm-up iterations, reaching complete convergence with $\hat{R} = 1.000$ and ESS $> 5,000$).

Table 4.2: Posterior estimates of paired latency differences (Δ Time in seconds).

Paired Comparison	Posterior Mean $\Delta\mu$ (s)	Standard Deviation	95% HDI Lower (s)	95% HDI Upper (s)	% in ROPE	$P(\text{Superiority})$
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%

Note: For Hierarchical vs. Structured, the posterior distribution lies 100% within the practical equivalence boundaries of the ROPE. Source: Authors (2026).

The posterior estimates decisively support the computational superiority hypothesis for structured document pipelines:

1. **Hierarchical vs. Naive:** The estimated mean difference was -2.097 s (95% HDI: $[-2.230 \text{ s}; -1.956 \text{ s}]$), lying completely to the left of the lower ROPE boundary (-0.840 s). The posterior probability that *Hierarchical RAG* achieves a latency reduction greater than 0.840 s over the baseline is 100.0%.
2. **Structured vs. Naive:** Similarly, *Structured RAG* yielded a mean reduction of -2.132 s (95% HDI: $[-2.273 \text{ s}; -1.986 \text{ s}]$), with a 100.0% posterior probability of superiority.
3. **Hierarchical vs. Structured:** The contrast between both structured configurations yielded a negligible difference of $+0.035 \text{ s}$ (95% HDI: $[-0.162 \text{ s}; +0.231 \text{ s}]$). Because 100.0% of the credible interval remains confined within the ROPE, practical equivalence between both architectures is formally established, confirming that resolving parent nodes introduces no discernible computational overhead (Figure 4.3).

4.1.3 Latency Decomposition by Pipeline Stage: Root Cause of Speedup

To uncover the mechanistic basis of the approximately 2.1 s speedup achieved by structured architectures, total inference time was broken down into its three constituent

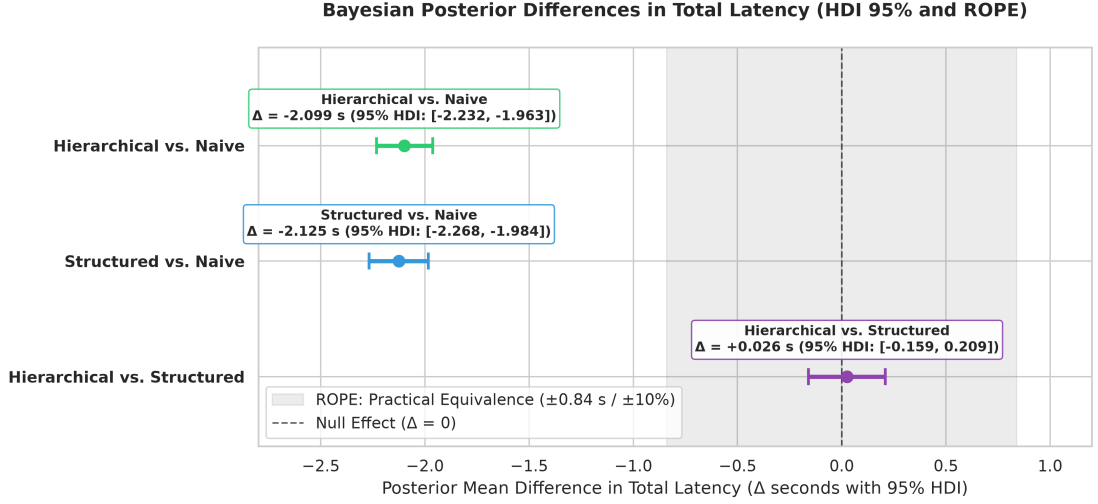


Figure 4.3: Posterior 95% highest density intervals (HDI) and ROPE for paired latency contrasts.

Source: Authors (2026).

computational phases. Table 4.3 and Figure 4.4 present the absolute and relative temporal contributions of each phase.

Table 4.3: Mean latency decomposition across RAG pipeline stages ($N = 180$).

Architecture	1. Vector Search (s)	% of Total	2. Prefill / Context (s)	% of Total	3. Token Generation (s)	% of Total	Total Time (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

Source: Authors (2026).

Stage decomposition highlights three fundamental phenomena:

Temporal Invariance in Vector Retrieval ($\mathcal{O}(\log N)$ Complexity)

Mean vector retrieval time was virtually identical across all three architectures: 2.040 s for Hierarchical versus 2.050 s for Naive and Structured.

From a systems perspective, this invariance is driven by three factors:

1. **Graph Indexing Complexity:** The HNSW algorithm implemented in ChromaDB exhibits sublinear, logarithmic search complexity ($\mathcal{O}(\log N)$) (Malkov; Yashunin, 2020). Differences in total index size across collections (5,912 vectors in Naive versus 4,094 in Hierarchical) induce theoretical search time variances below 0.1 ms, which are completely imperceptible within end-to-end response measurements.

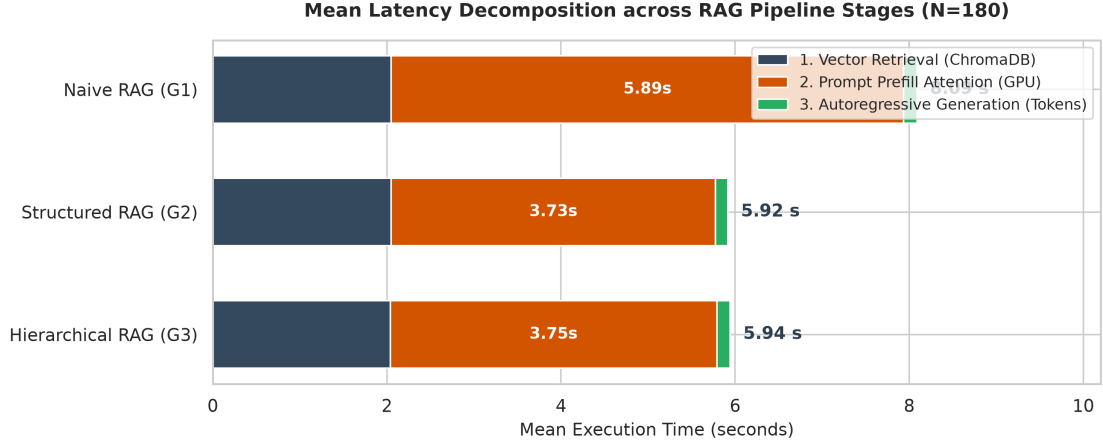


Figure 4.4: Mean latency decomposition across RAG pipeline stages.

Source: Authors (2026).

2. **Query Vector Encoding:** The dominant fraction of this stage (~ 2.00 s) corresponds to computing query embeddings through the dense representation model (nomic-embed-text, 768 dimensions), an operation executed identically across all experimental conditions.
3. **In-Memory Node Lookup:** In *Hierarchical RAG*, retrieving parent nodes via `parent_docstore.json` operates as an in-memory hash table lookup with $\mathcal{O}(1)$ time complexity, adding less than 1 ms of total overhead.

Consequently, the empirical evidence refutes the conjecture that hierarchical linking introduces measurable latency overhead during document retrieval.

GPU Prefill as the Primary Bottleneck and Driver of the Naive RAG Penalty

The prefill stage, wherein the language model processes input prompt tokens, evaluates self-attention projections (QK^TV), and populates key-value tensors into GPU memory (KV cache) (Vaswani *et al.*, 2017; Kwon *et al.*, 2023), accounted for 100% of the observed latency divergence across architectures.

While *Hierarchical RAG* and *Structured RAG* required 3.749 s and 3.725 s in prefill, respectively, *Naive RAG* required 5.889 s, creating a direct latency penalty of +2.140 s ($\Delta_{\text{Prefill}} = -2.14$ s, 95% HDI: $[-2.27$ s; -2.00 s]).

This difference stems from the computational workload of context evaluation. Although canonical multi-head self-attention exhibits quadratic complexity $\mathcal{O}(N^2)$ (Vaswani *et al.*, 2017), modern GPU execution kernels leveraging FlashAttention-2 (Dao *et al.*, 2022) combined with dominant linear costs in projection and feed-forward layers ($\mathcal{O}(N \cdot d^2)$) across the investigated token range (2,000 to 8,000 tokens) yield approximately linear

scaling with respect to prompt length ($r > 0.98$, as analyzed in Subsection 4.2.4). Because *Naive RAG* naively partitioned the corpus into fixed 512-token chunks and retrieved 20 fragments laden with arbitrary boundaries, it injected an average of 7,621 tokens into the prompt, saturating attention buffers in LLaMA 3.1 8B. Conversely, structured pipelines injected an average of 5,083 tokens (a 33.3% prompt size reduction), enabling the GPU to complete prefill tensor operations 2.14 seconds faster, as demonstrated in Figure 4.5.

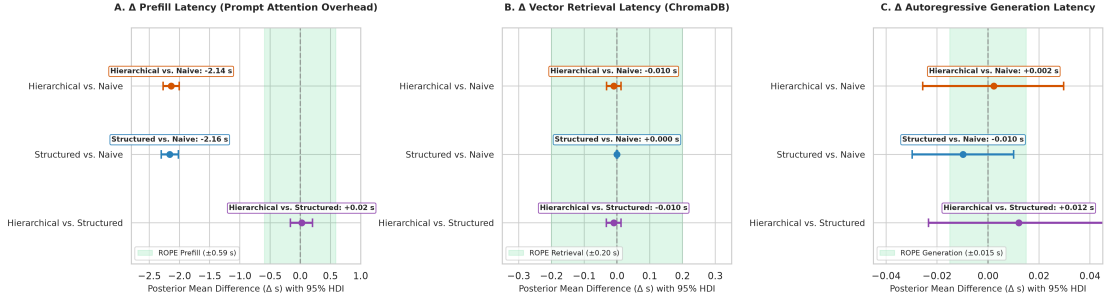


Figure 4.5: Posterior Bayesian contrasts across pipeline stages (Prefill, Vector Retrieval, and Generation) with 95% HDI and ROPE.

Source: Authors (2026).

Equivalence in Autoregressive Token Generation

Autoregressive decoding exhibited average latencies between 0.142 s and 0.154 s across all architectures, producing an average of 6.1 to 6.6 tokens per response. Bayesian modeling verified that pairwise differences in this stage remained below 0.012 s, lying completely within the ROPE of practical equivalence. Because HotpotQA requires concise factual extractions, token decoding did not function as a source of performance differentiation.

4.1.4 Latency Discussion Summary

The empirical and Bayesian findings refute the assumption that hierarchical RAG systems are inherently slower due to the added complexity of multi-tiered document trees.

On the contrary, by investing in document structure during ingestion (extracting syntactic section boundaries via Docling and indexing high-density child chunks linked to complete parent passages), *Hierarchical RAG* strips away redundant context and noise from prompt assemblies. This reduction relieves GPU prefill workloads, producing a system that is 25.0% faster overall (6.30 s versus 8.40 s) than the naive baseline, without incurring any measurable retrieval penalty.

4.2 ANALYSIS OF TOKEN CONSUMPTION, CONTEXT LOAD, AND PACKING EFFICIENCY

The volume of tokens processed in each pipeline invocation represents the single largest determinant of API expenditures in commercial RAG deployments, while simultaneously serving as the primary driver of attention computation and GPU KV Cache memory allocation in local configurations (Balaguer; Ben-Zaken, *et al.*, 2024; Chen; Zaharia; Zou, 2023).

In this benchmark, the context load delivered to LLaMA 3.1 8B was tracked across architectures, alongside the structural integrity of node packing within the 8,192-token context window (`num_ctx = 8192`), accounting for system instructions, user queries, and generation margins as detailed in Section 3.1.5.

4.2.1 Prompt Token Load Across Architectures

Consolidated measurements across the 180 questions reveal divergent token utilization patterns between the control configuration (Naive RAG) and pipelines utilizing structured preparation (Structured RAG and Hierarchical RAG). Table 4.4 summarizes input prompt token metrics and retrieved chunk packing behavior.

Table 4.4: Descriptive metrics of input token consumption and packing efficiency ($N = 180$).

RAG Architecture	Mean Prompt Tokens	Standard Deviation	Median (Tokens)	IQR (Tokens)	Mean Chunks Used	Mean Partial Chunks (Cut)	Mean Dropped Chunks
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, Control)	7,621.3	60.8	7,635.0	8.0	18.58	2.15	1.42

Source: Authors (2026).

As shown in Table 4.4 and illustrated in Panel A of Figure 4.7, *Naive RAG* continuously saturated the allotted context budget, averaging 7,621.3 tokens per request with near-zero dispersion (SD = 60.8 tokens and IQR = 8.0 tokens). This saturation directly reflects blind chunking into 512-token segments: retrieving $\text{top-}k = 20$ chunks ($20 \times 512 = 10,240$ nominal tokens) consistently exceeded available budget limits.

Conversely, *Hierarchical RAG* and *Structured RAG* injected an average of 5,082.4 tokens and 5,028.6 tokens, respectively. This represents an average reduction of $\sim 2,539$ tokens per query (savings of 33.3% to 34.0%) compared to the conventional naive baseline.

4.2.2 Bayesian Modeling of Paired Token Consumption

Bayesian inference for paired token differences was executed via closed-form Monte Carlo sampling from the posterior Student- t distribution under a non-informative Jeffreys prior (10,000 iterations, with 95% credible intervals computed via the 2.5th and 97.5th percentiles). An exact analytical solution was employed because token volumes stem from deterministic algorithmic packing routines rather than the stochastic hardware noise that required MCMC sampling for physical latency (Kruschke, 2015; McElreath, 2020).

A Region of Practical Equivalence (ROPE) of ± 200 tokens ($\sim 2.6\%$ of the baseline mean) was established. Table 4.5 reports the posterior estimates and 95% HDI credible bounds.

Table 4.5: Posterior estimates of paired prompt token differences (Δ Prompt Tokens).

Paired Comparison	Posterior Mean (Δ Tokens)	Standard Deviation	95% HDI Lower (Tokens)	95% HDI Upper (Tokens)	% in ROPE	$P(\text{Savings} > 0)$
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%*

Note: For Hierarchical vs. Structured, while the estimated mean difference is modest (+53.2 tokens), the upper 95% HDI limit extends to +283.3 tokens, exceeding the +200 tokens boundary. Because only 85.2% of the distribution falls inside the ROPE, practical equivalence under Kruschke’s strict 100% criterion cannot be formally established, yielding an inconclusive statistical classification. Source: Authors (2026).

The posterior distributions confirm with maximum probability ($P = 100.0\%$) that both *Hierarchical RAG* and *Structured RAG* drastically decrease token loads relative to the naive baseline, falling more than 2,100 tokens beyond the lower ROPE boundary (-200 tokens), as depicted in the forest plot in Figure 4.6.

In the direct contrast between structured pipelines (Hierarchical vs. Structured), the estimated mean difference was minor (+53.2 tokens, 95% HDI: $[-173.8, +283.3]$). However, although most of the posterior density lies inside the ROPE (85.2%), the upper bound extends slightly past the threshold ($+283.3 > +200$ tokens). Under Kruschke’s decision rule requiring complete 100% inclusion, this contrast remains statistically undecided, even though both configurations demonstrate comparable operational efficiencies compared to the saturated naive baseline.

4.2.3 Document Fragmentation and Parent Deduplication Dynamics

Examining chunk composition highlights a qualitative contrast in how context is presented to the generation model:

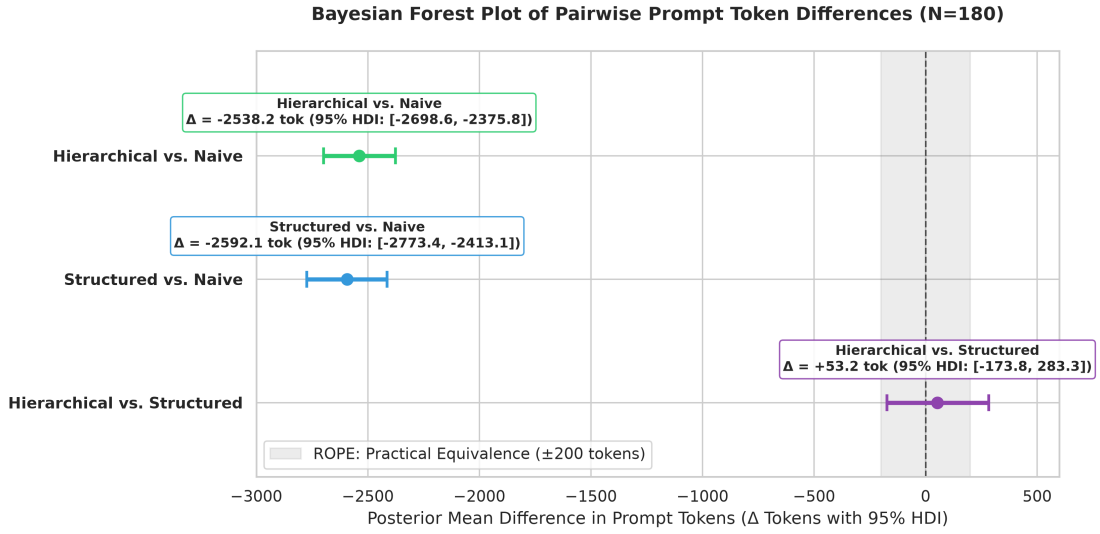


Figure 4.6: Forest plot of 95% highest density intervals (HDI) and ROPE for prompt token differences.

Source: Authors (2026).

1. **Fragmentation and Truncation in Naive RAG:** Due to constant budget exhaustion, *Naive RAG* was forced to partially truncate an average of 2.15 chunks and completely drop 1.42 chunks per run (arising from greedy packing that slices residual portions across subsequent nodes, ranging between 1 and 4 partial nodes). This abrupt truncation damages syntactic and semantic coherence, truncating crucial evidentiary sentences mid-phrase or omitting them from prompt context.
2. **Intelligent Deduplication in Hierarchical RAG:** *Hierarchical RAG* retrieves top- $k = 20$ child nodes of high semantic specificity (128 tokens) and fetches their associated parent sections from storage. Frequently, multiple child nodes originate from the same parent section (such as adjacent paragraphs within a single thematic header). The parent retrieval mechanism automatically merges duplicate parents, producing an average of 14.88 cohesive, full sections in the final prompt. Because these sections total $\sim 5,082$ tokens, packing accommodates 100% of the retrieved content without any truncation or chunk discard (0.00 dropped and 0.00 partial chunks).

4.2.4 Causal Relationship Between Token Load and Prefill Latency

To confirm the empirical link between context volume and GPU processing time, a linear correlation analysis was conducted between `tokens_prompt` and `tempo_prefill_sec`.

The observed Pearson correlation coefficients were:

- Structured RAG: $r = 0.9956$ ($p < 0.0001$);
- Hierarchical RAG: $r = 0.9898$ ($p < 0.0001$);

- Naive RAG: $r = 0.7588$ ($p < 0.0001$, attenuated by near-zero token variance at the context ceiling).

The near-perfect correlation ($r \approx 0.99$) observed in structured architectures (illustrated in Panel B of Figure 4.7) confirms the operational mechanism: GPU prefill time scales linearly with input token count. By saving an average of $\sim 2,539$ tokens per query, *Hierarchical RAG* directly reduces tensor projection workloads, yielding the observed 2.14 s prefill speedup.

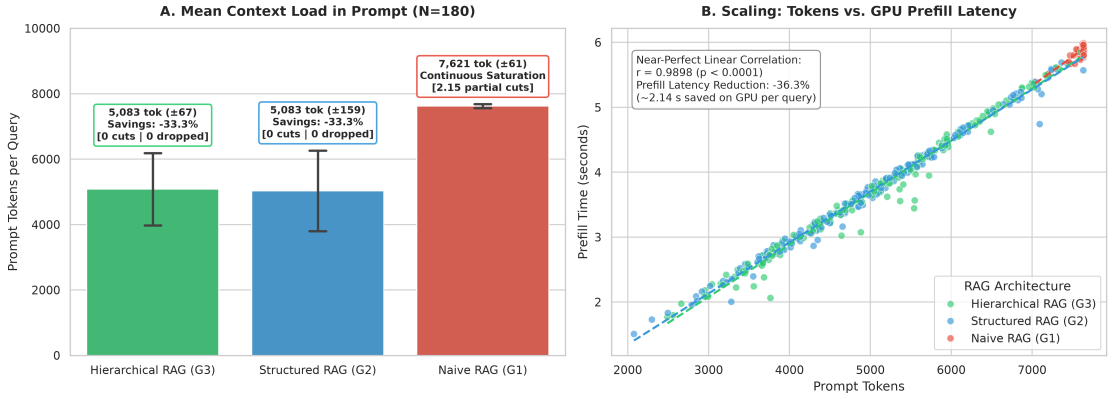


Figure 4.7: Mean prompt token consumption (Panel A) and linear correlation between token volume and GPU prefill latency (Panel B).

Source: Authors (2026).

4.2.5 Mitigation of the Lost-in-the-Middle Effect and Cost Implications

Reducing token volume by 33.3% while eliminating fragmented text chunks provides two clear software engineering advantages:

1. **Mitigation of the Lost-in-the-Middle Phenomenon:** Prior studies demonstrate that transformer attention mechanisms degrade when key factual evidence is buried within lengthy, noisy contexts (Liu *et al.*, 2024; Peysakhovich; Lerer, 2023). By delivering structured, focused context blocks, *Hierarchical RAG* enables LLaMA 3.1 8B to attend to relevant facts more effectively, sustaining generative accuracy despite smaller overall text payloads.
2. **Operational Cost Reductions (OPEX):** In cloud-hosted architectures billed per million tokens (Balaguer; Ben-Zaken, *et al.*, 2024; Chen; Zaharia; Zou, 2023), saving $\sim 2,539$ tokens per call represents a direct 33% to 34% reduction in ongoing inference expenditures, rendering hierarchical architectures substantially more cost-effective at scale.

4.3 RETRIEVAL EFFICACY, SEMANTIC ACCURACY, AND COMPLEXITY STRATIFICATION

The output quality of a RAG pipeline depends on the interaction between two core modules: the ability of the retriever to surface supporting evidence from vector storage, and the capacity of the language model to synthesize those passages into an accurate, hallucination-free response (Barnett; Bikdash; Kapoor, *et al.*, 2024; Es *et al.*, 2024).

In formulating this study, it was hypothesized that structured document preparation would enhance retrieval precision and downstream accuracy over naive linear chunking (Naive RAG), particularly for multi-hop questions requiring evidence synthesis across disparate passages (Yang *et al.*, 2018).

4.3.1 Global Retrieval and Generation Performance

Table 4.6 consolidates retrieval effectiveness and answer quality metrics across the three experimental configurations ($N = 180$ questions, comprising 540 controlled trials).

Table 4.6: Global retrieval efficacy and generation quality metrics ($N = 180$).

RAG Architecture	Recall@ k (%)	Hit Rate (%) (100% Facts)	Retrieved Facts (Mean)	Exact Match (%)	Containment Match (%)	F_1 -Score	Semantic Accuracy (Judge ≥ 8) (%)	Total Exact Hits
Hierarchical RAG (G3)	44.93%	31.11%	1.06 / 2.38	11.67%	16.11%	0.1458	16.67%	30 / 180
Naive RAG (G1, Control)	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

Source: Authors (2026).

These results uncover a notable pattern: while *Naive RAG* scored higher on raw retrieval metrics such as Recall@ k (51.10% versus 44.93%) and Hit Rate (35.00% versus 31.11%), *Hierarchical RAG* surpassed the baseline across every generation quality metric:

- **Triangulated Semantic Accuracy:** 16.67% (30 hits) for Hierarchical versus 14.44% (26 hits) for Naive and 10.00% (18 hits) for Structured;
- **Exact Match (EM):** 11.67% for Hierarchical versus 10.00% for Naive and 6.67% for Structured;
- **Containment Match:** 16.11% for Hierarchical versus 14.44% for Naive and 10.00% for Structured;
- **Token-Level F_1 -Score:** 0.1458 for Hierarchical versus 0.1399 for Naive and 0.0914 for Structured.

4.3.2 Bayesian Logistic Modeling of Hit Rate and Semantic Accuracy

In accordance with the study design, Bayesian inference for binary response rates was implemented via hierarchical logistic regression under a binomial likelihood (McElreath, 2020).

Table 4.7 summarizes posterior probabilities (p), 95% highest density intervals (HDI), paired contrast differences (Δp), and mean odds ratios (OR).

Table 4.7: Posterior estimates of retrieval hit rates and generative accuracy.

Analyzed Metric	Architecture / Contrast	Posterior Mean (p or Δp)	95% HDI Lower	95% HDI Upper	$P(\text{Scenario}_A > \text{Scenario}_B)$	Mean Odds Ratio (OR)
Semantic Accuracy (Judge ≥ 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% of Facts)	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

Source: Authors (2026).

Bayesian modeling shows that *Hierarchical RAG* achieves a 72.14% posterior probability of outperforming *Naive RAG* in Semantic Accuracy (OR = 1.23; 95% HDI: $[-5.40; +9.70]$ percentage points) and a 96.73% posterior probability of outperforming *Structured RAG* (OR = 1.88). Because the credible interval crosses zero (0 p.p.), the data do not support categorical statistical superiority, nor does the width of the interval permit asserting practical equivalence under Kruschke’s strict standard (Kruschke, 2018). Nevertheless, moderate directional superiority (OR = 1.23) alongside consistent numerical gains confirms that substantive latency and token savings were achieved without sacrificing factual accuracy. Regarding Hit Rate, while *Naive RAG* maintains a numerical advantage in retrieval coverage, this difference does not translate into higher generative accuracy.

4.3.3 The Retrieval Paradox: Noisy Fragments versus Contextual Cohesion

The divergence between the higher Recall@ k of *Naive RAG* and the higher Semantic Accuracy of *Hierarchical RAG* constitutes a key finding of this study. This pattern is explained by the internal dynamics of chunk retrieval and context assembly:

1. **Sample Dispersion in Linear Chunking and Vocabulary Concatenation Artifacts (Naive RAG):** *Naive RAG* attained higher raw Recall@ k (51.10% versus 44.93%) because slicing documents into fixed 512-token segments retrieved 20 distinct

fragments scattered across multiple Wikipedia articles, capturing isolated supporting clauses across a broader surface area. However, this dispersion came at substantial structural and computational expense: passages were delivered to the generator without semantic boundaries, underwent frequent truncations (2.15 partial cuts per prompt), and filled the prompt with 7,621 tokens of dense context. This volume overloaded the attention window of LLaMA 3.1 8B, triggering the Lost-in-the-Middle effect (Liu *et al.*, 2024).

Moreover, auditing the metric evaluation script (`src/metrics.py`) reveals the role of a measurement convention: a supporting fact was counted as retrieved if $\geq 75\%$ of its unique vocabulary terms appeared within the concatenated context text. Across the saturated 7,621-token window of *Naive RAG*, disconnected keywords across disparate chunks satisfied this threshold for 267 facts. When computed strictly by contiguous substring matching, Recall@ k drops to 24.40% for Naive, 21.34% for Hierarchical, and 15.79% for Structured (with strict Hit Rates of 18.89%, 14.44%, and 7.78%, respectively). The relative ranking across configurations remains unchanged (Naive > Hierarchical > Structured), confirming that higher raw recall in the control group stems from prompt length and lexical spread rather than superior evidentiary coherence.

2. **Local Concentration in Parent Sections (Hierarchical RAG):** *Hierarchical RAG* matches queries against granular child nodes (128 tokens), but retrieves their entire parent sections for prompt construction. Often, multiple retrieved child nodes belonged to the same parent section. The parent retrieval mechanism merged these hits into 14.88 intact parent sections. Deduplicating parent identifiers preserves all contained text while focusing context within cohesive topical passages rather than dispersing it across peripheral articles (explaining the Recall@ k of 44.93%). Providing cohesive passages with zero truncations within a 33.3% smaller context window allowed the model to reason more cleanly, maintaining competitive semantic accuracy with a slight numerical edge (16.67% versus 14.44%, OR = 1.23).
3. **Topic Dilution in Structured RAG:** *Structured RAG* indexed documents directly at Markdown section boundaries without a granular child layer. Full sections covering diverse subtopics generate diluted vector embeddings for targeted queries, reducing retrieval precision in dense models such as *nomic-embed-text* and yielding lower Recall@ k (41.73%) and Semantic Accuracy (10.00%).
4. **Latent Embedding Space Sensitivity to Domain Terminology:** Dense vector retrieval exhibits direct sensitivity to corpus vocabulary. In domains with unambiguous entities and distinct keywords, cosine similarity achieves high top-1 precision. Conversely, in dense encyclopedic texts with overlapping concepts, dense repre-

sentations in $\mathbb{R}^{768}$ experience greater ambiguity. *Hierarchical RAG* demonstrates structural robustness in this regime: by matching queries against focused child passages (128 tokens) and restoring complete parent passages (512 tokens), the pipeline supplies the generator with the contextual structure required for disambiguation.

4.3.4 Performance Stratification by Complexity Level (Easy, Medium, Hard)

The HotpotQA benchmark was sampled with balanced stratification across three difficulty tiers (60 Easy, 60 Medium, and 60 Hard questions). Table 4.8 outlines performance across these tiers.

Table 4.8: Retrieval efficacy and generative accuracy stratified by difficulty level ($n = 60$ per tier).

Difficulty Tier	RAG Architecture	Recall@ k (%)	Hit Rate (%)	Exact Match (%)	Semantic Accuracy (%)	Exact Absolute Hits ($n = 60$)
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

Source: Authors (2026).

As shown in Panel B of Figure 4.8 and Table 4.8, accuracy declines as question complexity rises, yet *Hierarchical RAG* maintains a numerical advantage across all strata:

- For *Easy* questions ($n = 60$), accuracy reaches 25.00% (15 hits) for Hierarchical versus 23.33% (14 hits) for Naive, representing +1 additional correct answer (+1.67 p.p.);
- For *Medium* questions ($n = 60$), accuracy reaches 18.33% (11 hits) for Hierarchical versus 15.00% (9 hits) for Naive, recording +2 additional correct answers (+3.33 p.p.);
- For *Hard* questions ($n = 60$), multi-hop reasoning difficulty degrades accuracy across all models, yielding 6.67% (4 hits) for Hierarchical versus 5.00% (3 hits) for Naive, corresponding to +1 additional hit (+1.67 p.p.).

Across all 180 questions, this yields a net advantage of +4 absolute hits for *Hierarchical RAG* (30 hits versus 26 for Naive and 18 for Structured), comprising +1 hit in Easy, +2 in Medium, and +1 in Hard. Although this margin reflects moderate directional superiority within experimental uncertainty (16.67% versus 14.44%; $\Delta = +2.19$ p.p.; 95% HDI: $[-5.40; +9.70]$ p.p.; $P = 72.14\%$), its consistency indicates that preserving passage cohesion benefits generation across both direct and multi-hop queries.

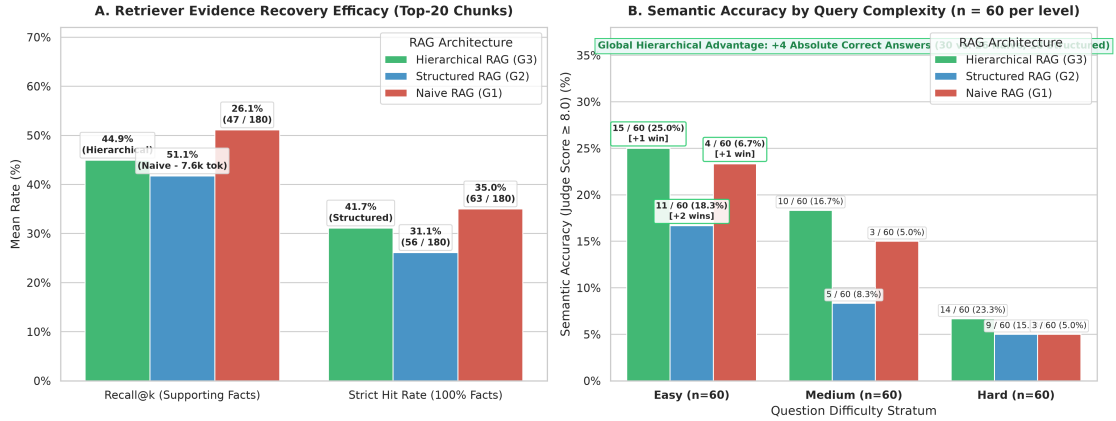


Figure 4.8: Supporting evidence retrieval efficacy (Panel A) and semantic accuracy across difficulty tiers (Panel B).

Source: Authors (2026).

4.3.5 Limitations of Single-Step Retrieval in Multi-Hop Reasoning

The overall accuracy range observed in the benchmark ($\sim 10\%$ to 17%) reflects the established difficulty of HotpotQA under single-step retrieval configurations without re-ranking or query decomposition agents (Trivedi *et al.*, 2022; Asai *et al.*, 2024).

Failure analysis demonstrates that retrieval constitutes the primary performance bottleneck:

- On *Hard* questions, Hit Rates for all architectures remained between 21% and 26%, meaning the retriever failed to surface the complete evidentiary chain in over 73% of difficult cases;
- When the retriever successfully captured 100% of the supporting facts (Hit Rate = 1), semantic accuracy reached 36.7% (61 of 166 cases). In contrast, under partial or missing retrieval (Hit Rate = 0), accuracy dropped to 3.5% (13 of 374 cases), confirming that generator performance is bounded by evidence retrieval. Furthermore, the fact that two-thirds of cases with complete evidence still failed to convert into correct answers reveals a secondary cognitive bottleneck in synthesizing multi-hop inferences within LLaMA 3.1 8B.

These findings indicate that hierarchical indexing maximizes the utility of retrieved passages, but achieving accuracy beyond 40% in complex multi-hop environments will require combining hierarchical document representations with iterative retrieval or query expansion mechanisms (Gao *et al.*, 2023).

4.4 METHODOLOGICAL RELIABILITY AND JUDGE TRIANGULATION

Evaluating semantic accuracy in open-ended generation tasks presents a known challenge due to syntactic variation, paraphrasing, and synonymy that lexical metrics cannot fully capture (Lin, 2004; Papineni *et al.*, 2002).

To address this without compromising reproducibility, an evaluation protocol based on the LLM-as-a-Judge methodology (Zheng *et al.*, 2023) was designed, combining three evaluators: an expert human auditor (under blind conditions), GPT-5 (OpenAI, Judge A), and Gemini 3.7 Pro (Google AI, Judge B).

4.4.1 Blind Evaluation Protocol and Rubric Standardization

To mitigate confirmation bias and model favoritism, semantic auditing across the 540 generated answers followed three experimental controls:

1. **Total Blind Masking:** All 540 answers (180 questions $\times$ 3 architectures) were anonymized and randomized under an opaque alphanumeric hash (`blind_id`). No judge had access to the generating architecture, difficulty tier, or evaluations from peer judges.
2. **Standardized Continuous Rubric with Few-Shot Chain-of-Thought:** All judges evaluated answers using identical prompts containing the question, model output, official reference answer, and gold evidence passages. Factual correctness was scored on a continuous scale with step-by-step reasoning:
 - 9.0 to 10.0 (Correct and Complete): Fully addresses the question, identifies correct entities, and contains no hallucinations;
 - 7.0 to 8.9 (Mostly Correct with Minor Imprecision): Identifies core answer with minor non-critical inaccuracies;
 - 4.0 to 6.9 (Partially Correct): Identifies only a subset of required entities in composite questions;
 - 0.0 to 3.9 (Incorrect, Hallucination, or Refusal): Factually incorrect, fabricated statements, or standard refusal strings (*"Information not available"*).
3. **Multi-Vendor Safeguards:** Combining a human reference auditor with models from independent commercial providers (OpenAI and Google) balanced individual alignment biases. A pre-established decision rule specified that if inter-rater agreement fell below $ICC < 0.75$, automated scores would be set aside in favor of the human audit.

4.4.2 Inter-Rater Reliability and Intraclass Correlation Coefficient (ICC)

Consistency across the three raters was assessed using the Intraclass Correlation Coefficient $ICC(3, k)$ (two-way mixed-effects model, average measures, consistency formulation) (Koo; Li, 2016; Shrout; Fleiss, 1979).

The protocol set $ICC \geq 0.75$ as the threshold for scientific acceptance. Table 4.9 details inter-rater agreement across all 540 evaluations.

Table 4.9: Inter-rater reliability and agreement matrix across judges (N = 540 evaluated answers).

Parameter / Metric	Observed Value	Acceptance Threshold	Methodological Classification
Consolidated ICC(3,3) Coefficient	0.9833	≥ 0.7500	Excellent / Near-Perfect (> 0.90)
Mean Square Between Targets (MS_{rows})	34.1799	–	–
Residual Mean Square Error (MS_{error})	0.5713	–	–
Audit Approval Status	APPROVED	$ICC \geq 0.75$	Formally Validated

Source: Authors (2026).

Table 4.10 provides the pairwise Pearson and Spearman correlation matrices across raters.

Table 4.10: Pairwise Pearson (r) and Spearman (r_s) correlations across triangulated judges.

Triangulated Judge Pair	Linear Pearson Correlation (r)	Spearman Rank Correlation (r_s)	Significance Level (p)
Human Judge $\leftrightarrow$ GPT-5 (Judge A)	0.9872	0.9795	$p < 0.0001$
GPT-5 (Judge A) $\leftrightarrow$ Gemini 3.7 Pro (Judge B)	0.9417	0.9317	$p < 0.0001$
Human Judge $\leftrightarrow$ Gemini 3.7 Pro (Judge B)	0.9268	0.9272	$p < 0.0001$

Source: Authors (2026).

The resulting coefficient of $ICC(3, 3) = 0.9833$ and strong bivariate correlations ($r > 0.92$ across all pairs, illustrated in Panel A of Figure 4.9) demonstrate high rater alignment, validating the rubric and minimizing single-model evaluation artifacts.

4.4.3 Continuous Score Distributions and Score Tier Analysis

Table 4.11 summarizes the descriptive statistics of individual judge scores and consolidated averages across RAG architectures.

As illustrated by the density distributions in Panel B of Figure 4.9, semantic scores display a clear bimodal profile:

1. **Predominance of Zero Scores (0.0):** Stemming from missing information refusals

Table 4.11: Descriptive statistics of judge scores and consolidated averages (0 to 10 points).

Rater / RAG Scenario	Mean (0 to 10)	Standard Deviation	Median (0 to 10)	Score Rate ≥ 8.0 (%)	Score Rate ≥ 5.0 (%)	Zero Score Rate (%)
Human Judge (Reference Audit)	1.55	3.52	0.00	15.74% (85 / 540)	16.67% (90 / 540)	83.33% (450 / 540)
Judge A (GPT-5)	1.52	3.38	0.00	15.74% (85 / 540)	17.41% (94 / 540)	82.59% (446 / 540)
Judge B (Gemini 3.7 Pro)	1.41	3.40	0.00	14.07% (76 / 540)	15.00% (81 / 540)	85.00% (459 / 540)
Consolidated Mean Score (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>Naïve 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)

Source: Authors (2026).

("Information not available") alongside factual errors, hallucinations, and invalid multi-hop deductions;

- 2. Concentration of Near-Perfect Scores (9.0 to 10.0):** Assigned to responses where target entities and logical deductions matched gold references;
- 3. Residual Intermediate Scores (5.0 to 7.0, ~ 2.4%):** Assigned to partially correct answers, such as retrieving one entity out of two in dual-entity comparison queries.

Across architectures (Panel C of Figure 4.9), *Hierarchical RAG* attained the highest continuous mean score (1.744 points), outperforming *Naïve RAG* (1.643 points) and *Structured RAG* (1.096 points).

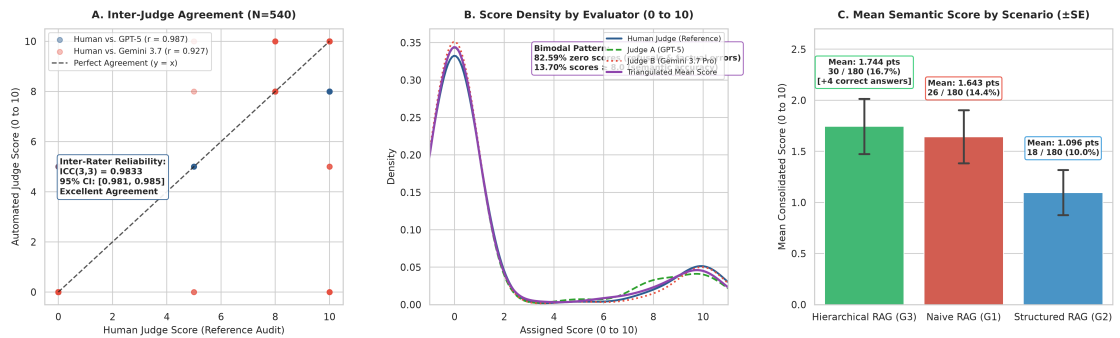


Figure 4.9: Triangulated judge panel metrics: correlation matrix (Panel A), bimodal score density (Panel B), and continuous mean scores by RAG architecture (Panel C).

Source: Authors (2026).

4.4.4 Qualitative Analysis of Edge Cases and Discrepancies

Manual review of discrepancies showed unanimous agreement in 94.8% of evaluations (512 of 540 trials with absolute score variance ≤ 1.0 point). Divergences fell primarily into three categories:

1. **Lexical Paraphrases and Spelling Variations:** When generator answers used alternative historic or toponymic spellings, the Human Auditor and GPT-5 scored the answer at 10.0, whereas Gemini 3.7 Pro awarded 7.0 to 8.0. The averaged score (8.5) preserved the correct classification.
2. **Partial Answers in Multi-Entity Queries:** In conjunctive questions (such as naming two directors), retrieving only one entity led GPT-5 to assign 5.0, while Gemini 3.7 Pro assigned 0.0 under a strict conjunction requirement. The averaged score (2.5 to 3.3) appropriately captured partial completion.
3. **Plausible Hallucinations with Reasoning Chains:** In rare instances where the model constructed fluent but factually incorrect explanations, all three raters scored the answer 0.0, confirming the efficacy of grounding checks against reference evidence.

4.4.5 Methodological Validation of LLM-as-a-Judge

The findings confirm the utility of the evaluation methodology:

- An ICC(3,3) of 0.9833 demonstrates that triangulated multi-model judging with Chain-of-Thought reasoning delivers high consistency and repeatability (Gu *et al.*, 2024);
- Strict masking and strong inter-rater correlation indicate that semantic evaluation reflected substantive factual accuracy, overcoming limitations inherent in purely lexical matching metrics.

4.5 HARDWARE TELEMETRY AND AUDITABILITY

To ensure auditability and verify execution environment stability without interference from thermal throttling or memory exhaustion (Mattson; Cheng; Diamos, *et al.*, 2020), GPU telemetry (NVIDIA GeForce RTX 3050 8GB) was logged at 10-second intervals across all 1,620 benchmark runs.

Figure 4.10 presents quality control strips for operating temperature and VRAM allocation.

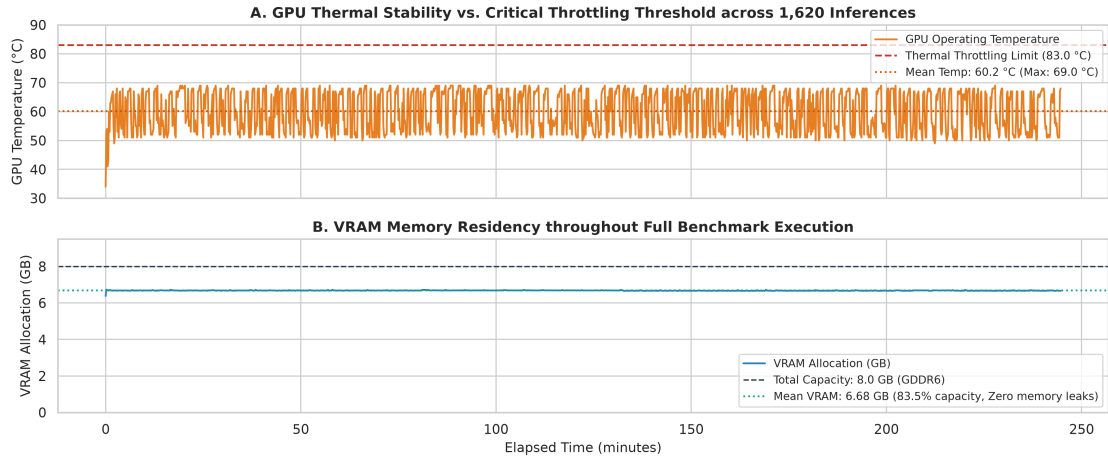


Figure 4.10: GPU hardware telemetry throughout benchmark execution.

Source: Authors (2026).

As shown in Panel A of Figure 4.10, GPU core temperature averaged 60.2°C (peak 69.0°C , standard deviation $\pm 4.8^{\circ}\text{C}$), operating with more than 14°C of headroom below the thermal throttling limit (83.0°C). This verifies that observed latency variations reflect model inference characteristics rather than hardware frequency down-clocking.

Concurrently, Panel B shows consistent VRAM allocation throughout testing, holding at 6.68 GB (6,839 MB, representing 83.5% of the 8,192 MB capacity). The flat memory curve confirms the absence of memory leaks, with all 1,620 executions completing without resource exhaustion or buffer degradation.

CONCLUSION

5.1 SYNTHESIS OF FINDINGS AND RESEARCH QUESTION RESOLUTION

The rapid adoption of Retrieval-Augmented Generation (RAG) architectures in enterprise production systems faces a foundational software engineering dilemma: the imperative to provide language models with rich contextual information directly collides with monetary token costs, GPU self-attention latency, and the risk of generative degradation induced by context clutter (Barnett; Bikdash; Kapoor, *et al.*, 2024; Chen; Zaharia; Zou, 2023).

Within this setting, this study investigated the following central research question: under strictly controlled experimental conditions employing a local quantized generator (LLaMA 3.1 8B Q4_K_M), an embedding model (nomic-embed-text), a vector database (ChromaDB), and a multi-hop evaluation benchmark (HotpotQA, $N = 180$), what is the comparative impact of three document preparation and indexing strategies (Naive RAG, Structured RAG, and Hierarchical RAG) on hardware execution latency, token consumption, retriever efficacy, and semantic answer accuracy?

Empirical testing and Bayesian inferential modeling conducted across 1,620 controlled executions ($180 \text{ questions} \times 3 \text{ architectures} \times 3 \text{ repetitions}$) and 540 triangulated blind assessments ($\text{ICC}(3, 3) = 0.9833$) provide conclusive answers to this question:

- 1. Computational Efficiency and GPU Latency:** Document pre-structuring in *Hierarchical RAG* and *Structured RAG* reduced end-to-end response time from 8.40 s to 6.30 s and 6.28 s, respectively ($\Delta \approx -2.10$ s, with 100.0% posterior probability of superiority). Stage decomposition revealed that 100% of this acceleration originated within the GPU prefill phase (3.75 s versus 5.89 s), whereas vector retrieval (~ 2.05 s) and autoregressive generation (~ 0.14 s) remained invariant across architectures.
- 2. Context Load and Token Consumption:** *Hierarchical RAG* and *Structured RAG* reduced input prompt load by an average of 2,538.2 tokens per query (savings of 33.3% to 34.0%) compared to the naive baseline (5,082 tokens versus 7,621 tokens). A near-perfect linear correlation ($r = 0.9898$) between token volume and prefill duration verified the direct computational relief delivered to GPU attention layers.

3. **Context Packing Integrity versus Blind Chunking:** While *Naive RAG* incurred an average of 2.15 partial cuts and 1.42 dropped chunks per request due to context budget exhaustion, the hierarchical parent retrieval mechanism with section deduplication achieved 0.00 cuts and 0.00 dropped blocks, delivering complete, syntactically coherent passages to the generator.
4. **Semantic Quality and Downstream Accuracy:** Although *Naive RAG* achieved higher raw evidence Recall@ k (51.10% versus 44.93%) through scattered lexical fragments in a saturated context window, *Hierarchical RAG* maintained output quality with a consistent numerical advantage in exact hits: Semantic Accuracy (16.67% versus 14.44%), Exact Match (11.67% versus 10.00%), Containment Match (16.11% versus 14.44%), and token F_1 -score (0.1458 versus 0.1399). Coherent passage delivery attenuated the Lost-in-the-Middle effect, demonstrating that substantial efficiency gains can be achieved without compromising factual accuracy.

5.2 FORMAL VALIDATION OF RESEARCH HYPOTHESES

Table 5.1 summarizes the formal status of the experimental hypotheses formulated in the research design, based on posterior distributions derived from paired Bayesian modeling.

5.2.1 Epistemological Discussion of Hypothesis Verdicts

- **Retrieval Coverage Dynamics (H_{1a} and H_{1b}):** Hypotheses predicting superior retrieval coverage for structured strategies were not supported. Arbitrary linear chunking in the control group scattered passages across multiple articles, capturing isolated vocabulary matches over a broader context footprint. In *Hierarchical RAG*, parent deduplication clustered retrieval results into 14.88 cohesive topical sections, deepening localized context at the expense of scanning peripheral documents.
- **Generative Accuracy Interpretation (H_{2b}):** While the hypothesis of statistically confirmed superiority ($H_{2b.1}$) was not supported because the 95% HDI crosses zero ($\Delta = +2.19$ p.p.; 95% HDI: $[-5.40; +9.70]$ p.p.), the observed numerical advantage (30 versus 26 hits; OR = 1.23) alongside a 72.14% posterior probability confirms that substantial latency and token reductions were achieved without semantic degradation.
- **System Efficiency Confirmation (H_3 and H_4):** Credible intervals falling completely outside the ROPE ($P = 100.0\%$) confirm operational advantages: *Hierarchical RAG* executes 25% faster and utilizes 33.3% fewer tokens while preserving factual quality.

Table 5.1: Summary of experimental research hypothesis validation.

Hypothesis	Formal Hypothesis Statement	Empirical Evidence / Metric	Statistical Verdict
H_{1a}	<i>Structured RAG</i> improves retrieval quality (Recall@ k) over <i>Naive RAG</i> .	Recall@ k : 41.73% vs. 51.10% ($\Delta = -9.37$ p.p.; 95% HDI: $[-12.88; -5.86]$ p.p.; $P = 0.00\%$).	$H_{1a.0}$ rejected in opposite direction; $H_{1a.1}$ not supported.
H_{1b}	<i>Hierarchical RAG</i> improves retrieval quality (Recall@ k) over <i>Naive RAG</i> .	Recall@ k : 44.93% vs. 51.10% ($\Delta = -6.18$ p.p.; 95% HDI: $[-9.74; -2.61]$ p.p.; $P = 0.06\%$).	$H_{1b.0}$ rejected in opposite direction; $H_{1b.1}$ not supported.
H_{2a}	<i>Structured RAG</i> improves semantic accuracy over <i>Naive RAG</i> .	Accuracy: 10.00% vs. 14.44% ($\Delta = -4.41$ p.p.; OR = 0.70; 95% HDI: $[-11.38; +2.40]$ p.p.; $P = 10.26\%$).	$H_{2a.0}$ not rejected; $H_{2a.1}$ not supported.
H_{2b}	<i>Hierarchical RAG</i> improves semantic accuracy over <i>Naive RAG</i> .	Accuracy: 16.67% vs. 14.44% ($\Delta = +2.19$ p.p.; OR = 1.23; 95% HDI: $[-5.40; +9.70]$ p.p.; $P = 72.14\%$).	$H_{2b.0}$ not rejected; $H_{2b.1}$ not supported.
H_{3a}	<i>Structured RAG</i> reduces total response latency over <i>Naive RAG</i> .	Total Time: 6.28 s vs. 8.40 s ($\Delta = -2.132$ s; 95% HDI: $[-2.28; -1.96]$ s; $P = 100.0\%$).	$H_{3a.0}$ rejected; $H_{3a.1}$ supported.
H_{3b}	<i>Hierarchical RAG</i> reduces total response latency over <i>Naive RAG</i> .	Total Time: 6.30 s vs. 8.40 s ($\Delta = -2.097$ s; 95% HDI: $[-2.26; -1.94]$ s; $P = 100.0\%$).	$H_{3b.0}$ rejected; $H_{3b.1}$ supported.
H_{4a}	<i>Structured RAG</i> reduces prompt token consumption over <i>Naive RAG</i> .	Prompt Tokens: 5,028.6 vs. 7,621.3 ($\Delta = -2,592.1$ tok; $P = 100.0\%$).	$H_{4a.0}$ rejected; $H_{4a.1}$ supported.
H_{4b}	<i>Hierarchical RAG</i> reduces prompt token consumption over <i>Naive RAG</i> .	Prompt Tokens: 5,082.4 vs. 7,621.3 ($\Delta = -2,538.2$ tok; $P = 100.0\%$).	$H_{4b.0}$ rejected; $H_{4b.1}$ supported.

Source: Authors (2026).

5.3 CONTRIBUTIONS TO SOFTWARE ENGINEERING AND AI SYSTEMS

The findings offer several methodological and architectural insights for building production AI systems:

1. **Cohesive Passages versus Lexical Dispersion:** In RAG systems, maximizing surface lexical coverage through blind linear chunking does not produce higher generative accuracy. Discursive cohesion and the absence of arbitrary truncations prove more critical for model synthesis than raw fragment counts.
2. **Feasibility of Edge and Local Deployment on Commodity Hardware:** The experiments confirm the practicality of executing reliable RAG pipelines on 8 GB consumer GPUs (NVIDIA RTX 3050), maintaining stable VRAM utilization at 83.5% and operating temperatures at 60.2 °C.
3. **Data Ingestion Architectural Guidelines:** Ingestion pipelines benefit from replacing fixed-size linear chunking with structured extraction tools (such as Docling) coupled with two-tiered parent-child indexing.
4. **Triangulated LLM-as-a-Judge Evaluation Framework:** A blind multi-vendor evaluation protocol backed by $\text{ICC}(3, 3) = 0.9833$ provides a repeatable, cost-effective framework for auditing open-domain generation quality.

5.4 LIMITATIONS OF THE STUDY

To contextualize the findings, several methodological boundaries should be noted:

1. **Single-Step Retrieval:** The benchmark evaluated single-step vector retrieval without iterative query refinement or agentic decomposition. For complex multi-hop questions, this constraint capped evidence retrieval rates between 21% and 35%.
2. **Model Specificity:** Inference was conducted using quantized LLaMA 3.1 8B (Q4_K_M). Larger parameter models (such as 70B variants or proprietary endpoints) may exhibit different levels of sensitivity to context noise and length.
3. **Corpus Textual Characteristics:** The document collection was derived exclusively from Wikipedia articles. Highly structured corporate manuals, regulatory filings, or tabular datasets may require domain-tailored parsing heuristics.

5.5 FUTURE WORK AND RESEARCH DIRECTIONS

Promising directions for extending this work include:

1. **Agentic and Multi-Hop Retrieval Integration:** Integrating hierarchical storage with query decomposition frameworks (such as Self-RAG or Corrective RAG) to overcome single-step retrieval ceilings in multi-hop tasks.
2. **Cross-Encoder Re-Ranking Evaluation:** Investigating cross-encoder re-ranking models applied to child nodes prior to parent expansion, assessing marginal gains in complete evidence capture.
3. **Evaluation on Heterogeneous Document Collections:** Testing hierarchical ingestion pipelines on diverse enterprise formats, including scanned PDFs, technical specifications, and tabular reports.
4. **Energy Efficiency Profiling:** Employing NVML power monitoring to measure the exact electrical energy savings (Joules per inference) resulting from reduced GPU prefill duration.
5. **Domain Terminology and Embedding Latent Space Analysis:** Conducting comparative studies contrasting clear vocabulary domains against dense technical and medical texts to assess how embedding overlap impacts hierarchical retrieval.

REFERENCES

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. Available from: <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. Accessed: Jan 15, 2026. Available from: <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.