RAG-NIMBF: A Non-Invasive, Modular Benchmarking Framework for Automated RAG Configuration Analysis

Niclas Hart*[,1] and Nicolas Weeger[1]

Ansbach University of Applied Sciences
n.hart18293@hs-ansbach.de, nicolas.weeger@hs-ansbach.de

Abstract. Retrieval-Augmented Generation (RAG) has become the dominant paradigm for grounding the outputs of large language models (LLM) in external knowledge. However, practitioners are faced with a vast configuration space comprising chunking, embedding, retrieval, reranking, prompting and generation models, and there is limited guidance on how these components interact. We present RAG-NIMBF, a modular benchmarking framework that attaches to existing RAG pipelines as a non-invasive evaluation layer. It evaluates the configuration space through controlled ablation with end-to-end experiment tracking and is publicly available via GitHub.[1] The evaluation combines LLM-based answer quality scores, retrieval correctness measures, and runtime indicators. This study validates the framework across multiple generation models on the SQuAD dataset. This process reveals non-obvious interactions and demonstrates how the experiments are tracked. The findings indicate that the retrieval balance of relevance and diversity substantially enhances the relevance and coverage of retrieved context in comparison to a conventional similarity search, with MMR raising nDCG@5 by +4 to +11 percentage points across the swept models. More detailed prompt instructions meaningfully increase how faithfully generated answers stick to the retrieved evidence, especially for capable and reasoning models (+10 to +13pp on Qwen3.6-27B, GPT-4o-mini and Qwen3.5-think, but negligible on the 20B generator). Larger generation models improve answer quality at the cost of higher latency, yet the local Qwen3.6-27B matches GPT-4o-mini at mean faithfulness 0.89, and the frontier reference does not dominate the smaller generators on RAG-relevant metrics. Overall, RAG-NIMBF transforms the configuration space of RAG into reproducible, comparable experiments. By attaching to existing production pipelines in a non-invasive way, it enables practitioners to systematically evaluate and improve deployed systems without disrupting existing workflows. This replaces ad hoc choices with evidence-based decisions.

Keywords: Retrieval-Augmented Generation · Benchmarking Framework · RAG Evaluation · Configuration Analysis

* Corresponding author

[1] https://github.com/aiengineeringblueprints/RAG-NIMBF

1 Introduction

Retrieval-Augmented Generation (RAG) [18] has emerged as the standard architecture for building knowledge-grounded question-answering systems. Rather than relying solely on parametric knowledge stored in model weights, RAG pipelines retrieve relevant documents from an external corpus and condition answer generation on the retrieved context. This approach reduces hallucination [32] and enables domain-specific applications without costly fine-tuning.

A production RAG pipeline involves multiple interacting components:

1. A chunking strategy splits source documents into retrievable segments.
2. An embedding model maps chunks and queries to a shared vector space.
3. A retrieval method selects the most relevant chunks.
4. An optional reranker re-orders them.
5. A prompt template frames the generation task.
6. A generation model produces the final answer [9].

Each component offers multiple design choices resulting in a large combinatorial configuration space. Even small changes in chunk size, retrieval strategy, or prompt wording can shift faithfulness scores by more than ten percentage points [6]. However, practitioners often select configurations based on intuition, vendor claims, or community defaults rather than systematic evaluations against their own data [8,31], and such defaults are highly sensitive to corpus and model [1,28]. For smaller organisations, building RAG on consumer-grade hardware rather than dedicated GPU clusters is the only economically viable path [33], so this study covers models that run on such hardware. Although several evaluation frameworks exist [8,31,30], their primary focus is on scoring a given pipeline rather than treating the configuration itself as the object of study. Systematic sweeps, component-interaction analysis, and reproducibility tracking are typically out of scope, and none of them are compatible with existing RAG pipelines.

These issues are addressed by RAG-NIMBF (RAG-Non-Invasive Modular Benchmarking Framework), a framework which is designed around three core principles:

1. Non-invasive integration: The framework wraps existing RAG components without modifying model or retriever internals. It connects to Ollama and OpenAI compatible endpoints (vLLM, TGI, ...) through provider routing, requiring only model identifier strings and access data.
2. Systematic configuration exploration: A declarative environment configuration specifies parameter ranges for all components. The framework generates the cartesian product and benchmarks every combination, producing and tracking structured results suitable for direct comparison.
3. Hybrid evaluation: Each run is assessed through a combination of answer-quality, retrieval, generation, and runtime metrics, so that configuration effects can be inspected from multiple complementary angles.

RAG-NIMBF is validated by benchmarking several configurations across different generation models spanning local, cloud-hosted, and API deployments

on a 100-question sample from the SQuAD dataset [26]. Experiment tracking is built into the framework itself, with every run automatically logged in MLflow. This includes parameters, metrics, artifacts and traces, so the results reported here are fully reproducible without additional bookkeeping.

The following design choices materialise in three concrete contributions:

1. A modular, non-invasive benchmarking framework for systematic RAG configuration analysis, with integrated MLflow tracking that makes every sweep reproducible by default.
2. A hybrid evaluation methodology combining answer-quality, retrieval, and generation metrics with runtime measurements.
3. An empirical study revealing non-obvious interactions between chunking, retrieval, prompting, and generation components, with explicit focus on models that fit on consumer-grade hardware so that smaller organisations can adopt the findings without dedicated AI infrastructure.

2 Related Work

Lewis et al. [18] introduced RAG by combining dense passage retrieval [14] with sequence-to-sequence generation. Subsequent systems have extended retrieval into pre-training, self-reflection, dynamic retrieval, graph-based reasoning and evidence checking [10,2,12,7,36]. Although surveys [9] demonstrate that output quality is affected by chunking, retrieval, and prompt design, they offer no tooling for the systematic comparison of configurations.

Several frameworks evaluate a single pipeline configuration. RAGAS [8] introduced LLM-based metrics (faithfulness, answer relevance, context precision/recall). ARES [31] uses prediction-powered inference without gold references.
RAGChecker [30] decomposes performance into retrieval and generation diagnostics. Chen et al. [6] benchmark LLMs on RAG but evaluate only model-level differences. Observability platforms such as LangSmith [16] and Langfuse [17] provide tracing and cost telemetry without Cartesian-product sweeps. However, none of these systems sweep the multi-component configuration space systematically.

A second class provides end-to-end RAG frameworks. LangChain [5] focuses on application composition rather than evaluation. FlashRAG [13] provides a modular toolkit comprising retrievers, generators, datasets, and a benchmark runner over preset configurations. However, it adopts an opinionated in-framework pipeline, meaning components must conform to its abstractions. Furthermore, evaluation couples neither IR, NLG, nor runtime indicators under a single non-invasive layer. DSPy [15] optimises prompt/few-shot choices within a configured pipeline rather than enumerating the Cartesian space. CRUD-RAG [22] contributes a Chinese-language benchmark over fixed pipelines.
BERGEN [27] benchmarks retrieval and reader modules under fixed generation pipelines, but does not consider chunking, prompts, or runtime. RAGPerf [19] profiles production-like latency and throughput but does not enumerate the configuration space or couple IR/NLG with runtime.

The individual components have been studied in isolation. The lost-in-the-middle effect [21], MMR [4], MTEB [23], and Sentence-BERT [29]. These studies examine one component at a time, leaving interaction effects unexplored. Similarly, RAG benchmarks are usually dataset-centric, such as SQuAD [26], rather than configuration-centric.

By contrast, RAG-NIMBF is designed as a non-invasive evaluation layer that wraps existing components using provider routing. It treats every pipeline stage as a swept dimension and fuses RAGAS-based quality, IR, NLG, and runtime metrics under MLflow-tracked runs, leaving the host pipeline intact.

3 Framework Architecture

RAG-NIMBF treats the configuration of a RAG pipeline, not the pipeline itself, as the object of study. Given a declarative specification of parameter ranges, it runs the cartesian product of all combinations and evaluates each run with a hybrid metric suite, leaving the host pipeline intact.

The design was guided by three principles:

1. Non-invasive integration: components are addressed through model-identifier strings and provider routing, so existing RAG setups attach without modifying model or retriever internals..
2. Systematic configuration exploration: an exhaustive cartesian sweep instead of surrogate-based search, so interaction effects across stages remain observable.
3. Hybrid evaluation: RAGAS, IR, NLG, and runtime metrics are logged jointly to MLflow, so no single score can be optimised in isolation.

RAG-NIMBF's pipeline proceeds through six stages, configuration, data loading, chunking, retrieval, generation, and evaluation, each implemented as an independent module. Figure 1 shows the overall data flow, a declarative configuration drives every downstream stage, intermediate artifacts (chunks, vector indices, retrieved contexts) are cached for reuse across configurations, and all parameters and metrics are logged to MLflow.

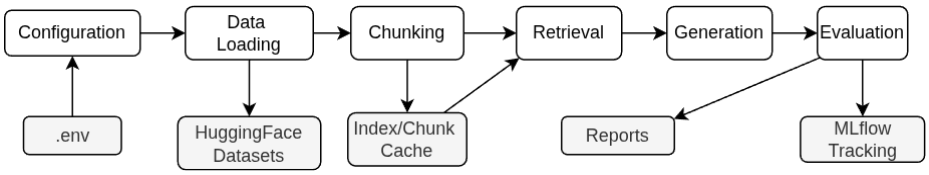

Fig. 1. Architecture of RAG-NIMBF.

3.1 Configuration and Data Loading

The framework reads all parameters from a single environment file. Users specify comma-separated lists for each sweepable parameter and the framework computes their cartesian product. The framework supports sweepable generation

models, chunking strategies, chunk sizes, overlap, retrieval method, top-k, prompt templates, HyDE, and reranking. We deliberately favour an exhaustive Cartesian sweep over more sample-efficient alternatives such as Bayesian optimisation or random search. Interaction effects across pipeline stages, for example chunk size against retrieval strategy or prompt template against model size, are the primary object of study. A surrogate-based optimiser would only reveal the best-performing point while leaving the interaction surface unobserved. The Cartesian product yields the full grid required for cross-configuration comparison and ablation. Model identifiers use provider:name syntax with automatic provider detection, e.g. ollama:qwen3:32b and openai:Qwen/Qwen3-32B-AWQ. A separate base URL can be configured per role, including generation LLM, critic LLM and embedding. This enables distributed setups and improves reproducibility by pinning each component to a fixed endpoint.

A registry-based adapter pattern maps dataset identifiers to loading functions. Each adapter converts a HuggingFace dataset into a unified {question, ground_truth, context} format. Built-in adapters cover SQuAD [26], RagBench, T2-RagBench, Ragas-WikiQA, and RagPerf-Wikipedia-NQ. For datasets with a shared corpus, documents are loaded once and reused across configurations.

3.2 Chunking

The framework supports three chunking strategies via LangChain text splitters [35]:

1. Recursive splitting with a separator hierarchy and a robust default for prose.
2. Fixed-size splitting using a token/character window with overlap and a deterministic baseline.
3. Semantic splitting using embedding-similarity breakpoints to preserve topical coherence.

Sentence-aware, markdown-aware, and late chunking were considered but not adopted due to high variance on noisy text, no structural markup in the corpus and compute-heavy long-context requirements respectively. For semantic chunking, chunk size and overlap are ignored since boundaries are determined by embedding similarity.

3.3 Retrieval and Indexing

Chunks are embedded and stored in ChromaDB or LanceDB. We chose these two over FAISS, Milvus, Qdrant, Weaviate, and pgvector for their local-first persistence, low operational cost, and sufficient scale for our evaluation. Vector stores are cached using content fingerprint keys, allowing index reuse across configurations that share the same chunking and embedding setup.

Retrieval supports similarity search (k nearest neighbours) and Maximal Marginal Relevance (MMR) [4], which balances relevance with diversity through a tunable λ parameter. An optional HyDE step generates a hypothetical answer

from the query and uses it as the retrieval input. An optional cross-encoder reranker re-scores retrieved documents and selects the top-n for generation. Sparse/hybrid retrieval (BM25, dense+sparse fusion) is out of scope.

3.4 Generation and Provider Routing

The framework routes requests to Ollama or OpenAI-compatible endpoints based on the model prefix. A wrapper normalises response formats for RAGAS compatibility. Answer post-processing strips <think> traces emitted by reasoning models before scoring, so that faithfulness reflects the final answer rather than intermediate chain-of-thought, keeping reasoning and non-reasoning models comparable under the same evaluation protocol.

3.5 Evaluation

Each configuration is evaluated through four complementary metric layers (Table 1).

Table 1. Metric definitions per evaluation layer (RAGAS layer from Es et al. [8]).

Layer	Metric	Definition
RAGAS	Faithfulness	$\lvert\{c_i \in \mathcal{C}_{\text{ans}} : c_i \text{ entailed by } \mathcal{C}_{\text{ctx}}\}\rvert / \lvert\mathcal{C}_{\text{ans}}\rvert$
	Context recall	$\lvert\text{GT attr. to } \mathcal{C}_{\text{ctx}}\rvert / \lvert\text{GT}\rvert$
	Semantic sim.	$\cos(\text{emb}(\text{ans}), \text{emb}(\text{GT}))$
IR	Hit@k	$\mathbb{1}[\lvert\mathcal{R}_k \cap \mathcal{G}\rvert \geq 1]$
	Recall@k	$\lvert\mathcal{R}_k \cap \mathcal{G}\rvert / \lvert\mathcal{G}\rvert$
	nDCG@k [11]	DCG@k/IDCG@k, $\text{DCG@}k = \sum_{i=1}^{k} \text{rel}_i / \log_2(i+1)$
NLG	BERTScore-F1, METEOR, ROUGE-L, BLEU [37,3,20,25]	surface-overlap vs. reference
Runtime	TTFT, TPS	time-to-first-token, tokens-per-second from server

Here, $\mathcal{C}_{\text{ans}}$ denotes atomic claims in the answer, $\mathcal{C}_{\text{ctx}}$ the retrieved context, $\mathcal{R}_k$ the top-k retrieved items, $\mathcal{G}$ the gold items, and rel_i the relevance grade of item i. The RAGAS layer is computed by a separate critic LLM (Qwen3.5-397B-A17B, non-thinking mode) and measures answer quality against the retrieved context. The IR layer measures the quality of the retrieval against the gold passages in SQuAD and is therefore independent of the critic. The NLG layer measures surface-form similarity against the reference answer. The runtime layer records latency and throughput directly from the inference server. Reading the four layers jointly prevents any single metric from being gamed.

All parameters, metrics, and artifacts are logged to MLflow with a reproducibility bundle capturing the full environment state.

4 Experimental Setup

4.1 Models

The experimental matrix pairs single-workstation local generators with two frontier API references so that configuration effects can be observed within a single-GPU budget and against an upper bound. Table 2 lists the six generation models, spanning dense transformers, mixture-of-experts (MoE) architectures, reasoning variants, and local plus API deployments. The local lineup targets the prosumer / single-accelerator tier that smaller organisations can realistically deploy without dedicated GPU clusters. The cloud APIs serve as upper-bound reference points.

Table 2. Generation models with parameter counts.

Model	Params	Deployment
Qwen3-32B-AWQ[1]	32B dense	Workstation
GPT-OSS-20B[2]	20B (3.6B active)	DGX Spark (GB10)
Qwen3.5-35B-A3B[3]	35B (3B active)	GPU-Cluster
Qwen3.6-27B-NVFP4[4]	27B dense	Workstation
GPT-4o-mini[5]	∼20-30B (est.)	GPU-Cluster
DeepSeek-V4-Pro[6]	862B	API

[1]Huggingface, `https://huggingface.co/Qwen/Qwen3-32B-AWQ`, accessed: Jul. 20, 2026
[2]Huggingface, `https://huggingface.co/openai/gpt-oss-20b`, accessed: Jul. 20, 2026
[3]Huggingface, `https://huggingface.co/Qwen/Qwen3.5-35B-A3B`, accessed: Jul. 20, 2026
[4]Huggingface, `https://huggingface.co/sakamakismile/Qwen3.6-27B-Text-NVFP4-MTP`, accessed: Jul. 20, 2026
[5]OpenAI, `https://platform.openai.com/docs/models/gpt-4o-mini`, accessed: Jul. 20, 2026
[6]Huggingface, `https://huggingface.co/deepseek-ai/DeepSeek-V4-Pro`, accessed: Jul. 20, 2026

The largest open-weight model available to us, Qwen3.5-397B-A17B, an MoE with 397B total and 17B active parameters, is deliberately excluded from the generation sweep and reserved for the RAGAS critic role. Using the strongest available open model as the judge keeps evaluation quality uniform across configurations and avoids confounding generator quality with judge weakness. Running the critic on the same H100 cluster in non-thinking mode holds the evaluation harness fixed while the generators vary. DeepSeek-V4-Pro is included as a frontier reference but supports only similarity retrieval over its API, so it appears in the corresponding subset of the configuration space only.

4.2 Embedding and Dataset

All configurations use nomic-embed-text:latest [24] (768-dim) as a fixed embedding backbone, an open, locally deployable model suitable for single-workstation benchmarking without external API cost. Holding the embedding model constant isolates the effects of chunking, retrieval strategy, prompt template, and generation model; embedding-model comparison is therefore out of scope.

We use SQuAD [26] because it is a widely used extractive question-answering benchmark with gold passage supervision. This makes it possible to compute retrieval metrics (Hit@k, nDCG@k, Recall@k) against a critic-independent ground truth while also evaluating generated answers against short reference answers. Its public availability and broad use in QA and RAG studies make the results easier to reproduce and compare to prior work.

4.3 Configuration Space

Table 3 summarizes the parameters swept in our experiments. The full grid contains $M \times C \times S \times O \times R \times T = 6 \times 2 \times 2 \times 2 \times 2 \times 2 = 192$ configurations. The empirical study reported here instantiates a deliberately smaller subset of that surface, which turns the experiment into a controlled ablation rather than a benchmark over every framework option.

Table 3. Configuration space.

Parameter	**Values**
LLM Models ($M = 6$)	See Table 2
Chunking Strategies ($C = 2$)	Recursive, Semantic
Chunk Sizes ($S = 2$)	500, 1000
Chunk Overlaps ($O = 2$)	100, 200
Retrieval Methods ($R = 2$)	Similarity, MMR
Prompt Templates ($T = 2$)	Concise, Detailed
Retrieval Top-k	12
Reranker / HyDE	Disabled
Critic model (outside the sweep)	Qwen3.5-397B-A17B (non-thinking)

4.4 Prompt Templates

Figure 2 shows the two prompt templates used in the experiments. The Concise template enforces a single-sentence factual answer with few-shot exemplars. The Detailed template removes the exemplars and asks for a full-sentence answer with supporting reasoning.

Concise

```
Answer the question using ONLY the
provided context.
RULES:
- Short, direct, factual (one
  sentence).
- No reasoning, no markdown.
- If absent: 'I cannot answer from
  the provided context.'
Examples:
Q: Who wrote the music for Star
     Wars?
A: John Williams.
```

Detailed

```
Answer the question based only on
the provided context.
Provide a clear, complete answer
in full sentences.
Include relevant calculations or
reasoning when applicable. Do not
start with 'Based on the provided
context'. Answer directly.

Human turn (both):
Context: {context}
Question: {question}
Answer:
```

Fig. 2. The two prompt templates.

4.5 Evaluation Protocol

RAGAS metrics (faithfulness, context recall, semantic similarity) are computed using Qwen3.5-397B-A17B as the critic model with a maximum of 4096 critic tokens per evaluation call. IR metrics (Hit@k, nDCG@k, Recall@k for $k \in \{1, 3, 5\}$) are computed against gold document labels provided by the SQuAD dataset. NLG metrics use RoBERTa-large for BERTScore [37] and standard implementations for METEOR [3], ROUGE-L [20], and BLEU [25]. All metrics are reported as means over the 100-question sample, with per-question values retained for the bootstrap and significance analyses reported in Section 5.

5 Results

All metric values are means over 100 SQuAD questions per configuration. We organise findings into three research questions covering generation-model comparison, component effects and interactions, and runtime-quality tradeoffs. The results are preliminary and represent a subset of the planned evaluation.

5.1 RQ1: How Do Generation Models Compare?

Table 4 aggregates results across all configurations for each model.

Three models share the top mean faithfulness at 0.89 (GPT-4o-mini, Qwen3.6-27B, and the reasoning-only Qwen3.5-think at 0.88), with Qwen3-32B and GPT-OSS-20B trailing at 0.83 and 0.77. The best single configuration reaches up to 0.96 faithfulness (GPT-4o-mini), and 0.95 is attained by Qwen3.5-think and Qwen3.6-27B, showing that strong grounding is attainable across architectures and parameter budgets. DeepSeek-V4-Pro leads METEOR at 0.56 but is restricted to similarity retrieval and therefore appears in the corresponding subset of the configuration space only. Among the local models, Qwen3.6-27B matches GPT-4o-mini on mean faithfulness (0.89). The faithfulness ranking is robust within ± 0.01 based on a 95% bootstrap CI over 2000 resamples.

Retrieval metrics are reported separately because they are generator-invariant by construction. Hit@k, nDCG@k, and Recall@k are pure functions of the retrieved chunk set and the gold document, neither of which depends on the generator when the embedding model and retrieval configuration are held fixed. At fixed (*chunking*, *retrieval*, *top-k*) cells, nDCG@5 varies by at most 0.07 across generators, driven by per-run question-subset differences. Aggregated across all runs, the global retrieval quality is nDCG@5 = 0.75, Hit@5 = 0.86, Recall@5 = 0.69. Per-configuration values are reported in Section 5.2.

Table 4. Model comparison: mean and (best) across all configurations. Retrieval metrics omitted from per-model rows because generator-invariant by construction; global retrieval quality reported in the final row.

Model	**Faith. ↑**	**BS-F1 ↑**	**METEOR ↑**
Qwen3.5-think	0.88 (0.95)	0.86 (0.93)	0.49 (0.60)
GPT-4o-mini	0.89 (0.96)	0.87 (0.93)	0.49 (0.59)
DeepSeek-V4-Pro	0.88 (0.94)	0.91 (0.92)	0.56 (0.66)
Qwen3.6-27B	0.89 (0.95)	0.89 (0.95)	0.51 (0.61)
Qwen3-32B	0.83 (0.92)	0.91 (0.97)	0.48 (0.60)
GPT-OSS-20B	0.77 (0.84)	0.90 (0.96)	0.46 (0.60)
Retrieval quality (all 222 non-critic runs, generator-invariant)			
nDCG@5 / Hit@5 / Recall@5	0.75	0.86	0.69

5.2 RQ2: How Do Retrieval, Prompting, and Chunking Affect Quality?

The retrieval and prompt comparisons below use the five-model subset with full coverage of the swept configuration space. Table 5 compares similarity search and MMR on this subset. MMR improves nDCG@5 by +4 to +11pp by reducing redundancy in the retrieved set, with the largest gains on GPT-4o-mini (+11pp, 0.68 → 0.79) and Qwen3.5-think (+10pp, 0.70 → 0.80), and the smallest on GPT-OSS-20B (+4pp). The faithfulness effect reverses sign with model size, the two smaller generators Qwen3-32B (+5pp) and GPT-OSS-20B (+6pp) gain faithfulness under MMR, while GPT-4o-mini (−4pp), Qwen3.5-think (−2pp) and Qwen3.6-27B (−1pp) lose some. The smaller generators benefit from the broader evidence mix that MMR provides, whereas the stronger generators already ground well on the most similar passage and are diluted by the additional context.

Table 5. Retrieval strategy comparison: mean over chunk sizes, overlaps, and templates.

Model	Strategy	Faith.	nDCG@5	BS-F1	METEOR
Qwen3-32B	Sim.	0.80	0.71	0.92	0.51
	MMR	0.85	0.81	0.89	0.47
GPT-OSS-20B	Sim.	0.74	0.71	0.92	0.46
	MMR	0.80	0.75	0.89	0.45
GPT-4o-mini	Sim.	0.90	0.68	0.88	0.50
	MMR	0.86	0.79	0.82	0.47
Qwen3.6-27B	Sim.	0.90	0.67	0.90	0.52
	MMR	0.89	0.74	0.88	0.50
Qwen3.5-think	Sim.	0.88	0.70	0.85	0.47
	MMR	0.86	0.80	0.82	0.47

Table 6 shows the Detailed template raises faithfulness by +10 to +13pp on the cloud/large generators and the reasoning model (Qwen3.5-think +13pp, GPT-4o-mini +11pp, Qwen3.6-27B +10pp), while Qwen3-32B gains +7pp and GPT-OSS-20B essentially does not move (+1pp). The faithfulness gain trades off against METEOR (10-22pp drop with Detailed) because detailed, evidence-grounded answers diverge from the brief SQuAD references. BERTScore F1 is largely unaffected (0-4pp), confirming semantic stability. This lexical-vs-semantic divergence illustrates why multi-metric evaluation is necessary. Optimising METEOR alone would misleadingly favour under-specified answers.

Four combinations, namely 500/100, 500/200, 1000/100, and 1000/200, were tested on Qwen3-32B with recursive chunking. Effects are modest, faithfulness spans 0.70-0.87 and nDCG@5 spans 0.73-0.77 across the four settings. Smaller chunks of 500 characters yield slightly higher nDCG@5 (mean 0.76) than 1000-character chunks (mean 0.74) via finer granularity, while larger chunks marginally raise faithfulness at overlap 200 (0.87) via broader grounding. Semantic chunking yields identical results across all size/overlap combinations, since boundaries follow embedding-similarity breakpoints rather than character counts.

Table 6. Prompt template comparison: mean over all chunk sizes, overlaps, and retrieval strategies.

Model	Template	Faith.	nDCG@5	BS-F1	METEOR
Qwen3-32B	Concise	0.80	0.74	0.89	0.58
	Detailed	0.87	0.74	0.91	0.41
GPT-OSS-20B	Concise	0.76	0.74	0.91	0.56
	Detailed	0.77	0.74	0.91	0.34
GPT-4o-mini	Concise	0.83	0.75	0.83	0.53
	Detailed	0.94	0.75	0.85	0.42
Qwen3.6-27B	Concise	0.84	0.76	0.84	0.55
	Detailed	0.94	0.75	0.88	0.43
Qwen3.5-think	Concise	0.81	0.74	0.81	0.52
	Detailed	0.94	0.75	0.85	0.42

5.3 RQ3: Performance–Quality Tradeoffs

Table 7 reports the raw inference measurements for 15 prompts and a maximum of 1024 output tokens, for both sequential and parallel processing. DeepSeek-V4-Pro is excluded because its third-party-API latency is dominated by network and is therefore not comparable to the other checkpoints on identical hardware.

Qwen3.6-27B-NVFP4 leads sequential throughput (127.5 TPS) and parallel throughput among the locals (70.5 TPS). Figure 3 plots throughput against faithfulness. Qwen3.6-27B-NVFP4 anchors the high-throughput end, GPT-4o-mini and Qwen3.5-think the high-quality end. GPT-OSS-20B sits off the quality-speed trade-off.

Table 7. Runtime measurements (N=15, max 1024 output tokens). Seq.=sequential, Par.=parallel, TPS=tokens per second, TTFT=time-to-first-token.

Model	Seq. t (s)	Par. t (s)	Seq. TPS	Par. TPS	TTFT (s)
GPT-OSS-20B	77.09	44.56	53.9	22.4	0.47
Qwen3-32B-AWQ	87.13	22.23	58.9	45.4	0.30
GPT-4o-mini	54.93	28.76	79.4	59.1	0.45
Qwen3.5-think	80.19	26.43	78.8	68.5	3.50
Qwen3.6-27B-NVFP4	38.59	24.80	127.5	70.5	1.86

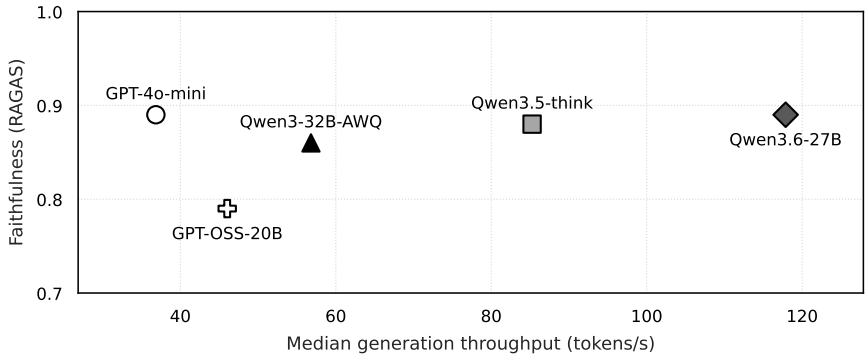

Fig. 3. Faithfulness vs. median generation throughput.

6 Discussion

The primary contribution of this work is RAG-NIMBF, a non-invasive benchmarking layer for systematically evaluating configurations across the RAG configuration space. Empirical findings suggest that the framework can help identify meaningful effects across pipeline stages, offering insights into the behaviour of different RAG configurations on consumer-grade hardware.

6.1 What the Framework Reveals

As RAG-NIMBF sweeps through every stage of the pipeline rather than fixing any component, the relative weight of each lever becomes apparent. The retrieval strategy and prompt wording can increase faithfulness and nDCG@5 by up to 11 percentage points, whereas the chunk size and overlap stay within 2-5 points. Interactions matter too, detailed prompts have almost no effect on GPT-OSS-20B but add +11 points on GPT-4o-mini. MMR reverses its faithfulness effect across model sizes.

The metrics also need to be considered together. Detailed prompts increase faithfulness but lower METEOR by 10-22 points, because SQuAD references are short and evidence-based answers differ in surface form while staying semantically aligned.

6.2 Recommendations

At fixed compute, the levers fall out of the sweep in a clear order. Prompting is the lowest-integration lever on capable models. Retrieval comes next, as measured by MMR's nDCG@5 gains. Chunk size and overlap have smaller effects. Once these are set, the generator can be scaled appropriately. In our sweep, Qwen3.6-27B is a reasonable substitute for the tested cloud APIs.

6.3 Limitations

The sweep leaves reranking, HyDE, and embedding-model comparisons out, holds top-k and the MMR λ fixed, and is as of now only evaluated on the SQuAD dataset. SQuAD is short, extractive, mostly single-passage QA, results may not transfer to long-document, multi-hop, noisy, or enterprise corpora. All quality assessments are automated and therefore hold relative to the RAGAS critic. The prompt templates were authored once and applied uniformly. It is not yet clear if different model families benefit from prompt structures that are structurally different.

7 Conclusion

This paper presents RAG-NIMBF, a non-invasive, modular benchmarking framework for systematic RAG configuration analysis. Across the swept SQuAD configurations, prompt specificity had the strongest effect on faithfulness, MMR improved retrieval coverage, and chunk-size effects were secondary. The results show why RAG configurations must be evaluated with multiple metrics: lexical scores, semantic similarity, retrieval quality, and runtime can move in different directions. Runtime measurements indicate that Qwen3.6-27B-NVFP4 can match GPT-4o-mini on mean faithfulness at lower latency, while the frontier reference model does not dominate on RAG-relevant metrics. Future work will extend the sweep to additional datasets, reranking, HyDE, and embedding models, and align RAG-NIMBF with on-premises enterprise RAG architectures such as the Weeger et al. blueprint [34].

8 Disclosure of Interests

The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Ammar, A., Koubaa, A., Nacar, O., Boulila, W.: Optimizing retrieval-augmented generation: Analysis of hyperparameter impact on performance and efficiency. arXiv preprint arXiv:2505.08445 (2025)
2. Asai, A., Wu, Z., Wang, Y., Sil, A., Hajishirzi, H.: Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In: International Conference on Learning Representations (2024)
3. Banerjee, S., Lavie, A.: METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In: Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization. pp. 65–72 (2005)
4. Carbonell, J., Goldstein, J.: The use of MMR, diversity-based reranking for reordering documents and producing summaries. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 335–336 (1998)

5. Chase, H.: LangChain. https://github.com/langchain-ai/langchain (2022)
6. Chen, J., Lin, H., Han, X., Sun, L.: Benchmarking large language models in retrieval-augmented generation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38 (2024), arXiv:2309.01431
7. Edge, D., Trinh, H., Cheng, N., Bradley, J., Chao, A., et al.: From local to global: A graph RAG approach to query-focused summarization. arXiv preprint arXiv:2404.16130 (2024)
8. Es, S., James, J., Espinosa-Anke, L., Schockaert, S.: RAGAS: Automated evaluation of retrieval augmented generation. arXiv preprint arXiv:2309.15217 (2023)
9. Gao, Y., Xiong, Y., Dixit, A., Gao, X., Liu, K., et al.: Retrieval-augmented generation for large language models: A survey. arXiv preprint arXiv:2312.10997 (2024)
10. Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.W.: REALM: Retrieval-augmented language model pre-training. In: International Conference on Machine Learning. pp. 3929–3938 (2020)
11. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems **20**(4), 422–446 (2002)
12. Jiang, Z., Xu, F.F., Gao, L., Sun, Z., Liu, Q., Yang, Y., Neubig, G.: Active retrieval augmented generation. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (2023), arXiv:2305.06983
13. Jin, J., Zhu, Y., Dong, G., Zhang, H., et al.: FlashRAG: A modular toolkit for efficient retrieval-augmented generation research. arXiv preprint arXiv:2405.13576 (2024)
14. Karpukhin, V., Öguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing. pp. 6769–6781 (2020)
15. Khattab, O., Singhvi, A., Maheshwari, P., Zhang, D., Santhanam, K., Vardhamanan, S., et al.: DSPy: Compiling declarative language model calls into self-improving pipelines. arXiv preprint arXiv:2310.03714 (2023)
16. LangChain Inc.: LangSmith: Tracing and evaluation for LLM applications. https://docs.smith.langchain.com (2024), software, accessed 2026-06
17. Langfuse GmbH: Langfuse: Open-source LLM engineering platform. https://langfuse.com (2024), software, accessed 2026-06
18. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive NLP tasks. In: Advances in Neural Information Processing Systems. vol. 33, pp. 9459–9474 (2020)
19. Li, S., Zhou, Y., Xu, Y., Chen, K., Waddington, D., Sundararaman, S., Franke, H., Huang, J.: RAGPerf: An end-to-end benchmarking framework for retrieval-augmented generation systems. arXiv preprint arXiv:2603.10765 (2026)
20. Lin, C.Y.: ROUGE: A package for automatic evaluation of summaries. In: Text Summarization Branches Out. pp. 74–81 (2004)
21. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. Transactions of the Association for Computational Linguistics **12**, 157–173 (2024), arXiv:2307.03172
22. Lu, Z., Liu, P., Tang, Z., Yang, Y., Li, W., et al.: CRUD-RAG: A comprehensive chinese benchmark for retrieval-augmented generation. arXiv preprint arXiv:2401.17043 (2024)
23. Muennighoff, N., Tazi, N., Magne, L., Reimers, N.: MTEB: Massive text embedding benchmark. arXiv preprint arXiv:2210.07316 (2022)

24. Nomic AI: nomic-embed-text: A high-performance open embedding model. Technical Report (2024), https://huggingface.co/nomic-ai/nomic-embed-text-v1.5
25. Papineni, K., Roukos, S., Ward, T., Zhu, W.J.: BLEU: A method for automatic evaluation of machine translation. Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics pp. 311–318 (2002)
26. Rajpurkar, P., Zhang, J., Lopyrev, K., Liang, P.: SQuAD: 100,000+ questions for machine comprehension of text. In: Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing. pp. 2383–2392 (2016)
27. Rau, D., Déjean, H., Chirkova, N., Formal, T., Wang, S., Nikoulina, V., Clinchant, S.: BERGEN: A benchmarking library for retrieval-augmented generation. arXiv preprint arXiv:2407.01102 (2024)
28. Ray, S., Pan, R., Gu, Z., Du, K., Feng, S., Ananthanarayanan, G., Netravali, R., Jiang, J.: RAGServe: Fast quality-aware RAG systems with configuration adaptation. arXiv preprint arXiv:2412.10543 (2024)
29. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using siamese BERT-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. pp. 3982–3992 (2019)
30. Ru, D., Qiu, L., Huang, X., Feng, J., Wang, Q., et al.: RAGChecker: A fine-grained framework for diagnosing retrieval-augmented generation. arXiv preprint arXiv:2408.08067 (2024)
31. Saad-Falcon, J., Khramtsova, O., Lin, M.O., Cristian, A.R., Khashabi, D.: ARES: An automated evaluation framework for retrieval-augmented generation systems. arXiv preprint arXiv:2311.09476 (2024)
32. Shuster, K., Poff, S., Chen, M., Kiela, D., Weston, J.: Retrieval augmentation reduces hallucination in conversation. In: Findings of the Association for Computational Linguistics: EMNLP 2021. pp. 3784–3803 (2021)
33. Weeger, N., Stiehl, A., von Kistowski, J., Geißelsöder, S., Uhl, C.: Towards practicable machine learning development using AI engineering blueprints. arXiv preprint arXiv:2504.06391 (2025)
34. Weeger, N., Winkler, J., Stiehl, A., von Kistowski, J., Uhl, C., Geißelsöder, S.: AI engineering blueprint for on-premises retrieval-augmented generation systems. arXiv preprint arXiv:2604.01395 (2026). https://doi.org/10.48550/arXiv.2604.01395
35. Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Cao, C., Jernite, Y., Plu, J., Xu, C., Le Scao, T., Gugger, S., Drame, M., Lhoest, Q., Rush, A.M.: Transformers: State-of-the-art natural language processing. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations. pp. 38–45 (2020)
36. Yu, W., Zhang, H., Zhang, X., Liu, T.: Chain-of-note: Enhancing robustness in retrieval-augmented language models. In: Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (2024), arXiv:2311.09210
37. Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q., Artzi, Y.: BERTScore: Evaluating text generation with BERT. In: International Conference on Learning Representations (2020)