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

Niclas Hart, Nicolas Weeger Hochschule für Angewandte Wissenschaften Ansbach

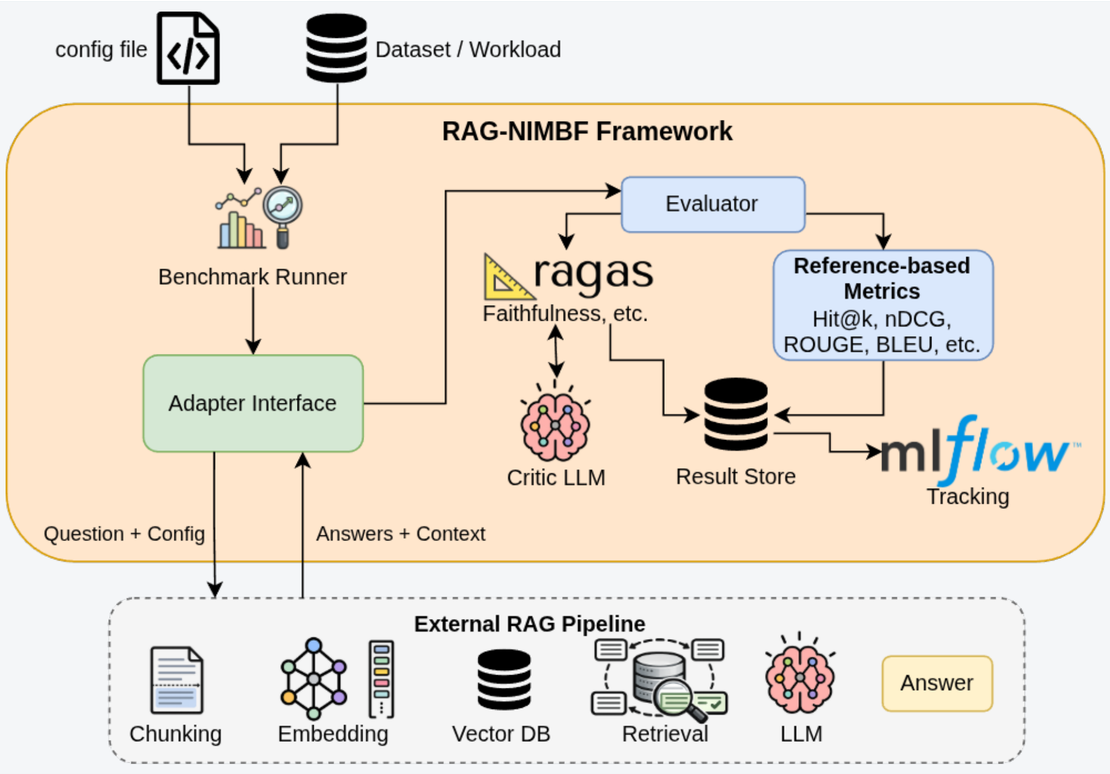
HOCHSCHULE ANSBACH

Problem and Goal

RAG pipelines expose many interacting components, chunking, retrieval, prompting, reranking, model, yet teams usually tune components in isolation. Interactions dominate: a prompt that helps one model can hurt another, MMR can trade faithfulness for diversity, and a faster generator may beat a larger one once retrieval is fixed. There is no single "best" RAG system, only best configurations for a given dataset, hardware, and quality target.

Goal: RAG-NIMBF benchmarks complete configurations across these axes so quality, runtime, and reproducibility can be compared without rewriting the pipeline per experiment.

Pipeline



What RAG-NIMBF Contributes

Non-invasive: the RAG pipeline is the system under test and is never modified. The framework only feeds configs, observes outputs, records evidence.

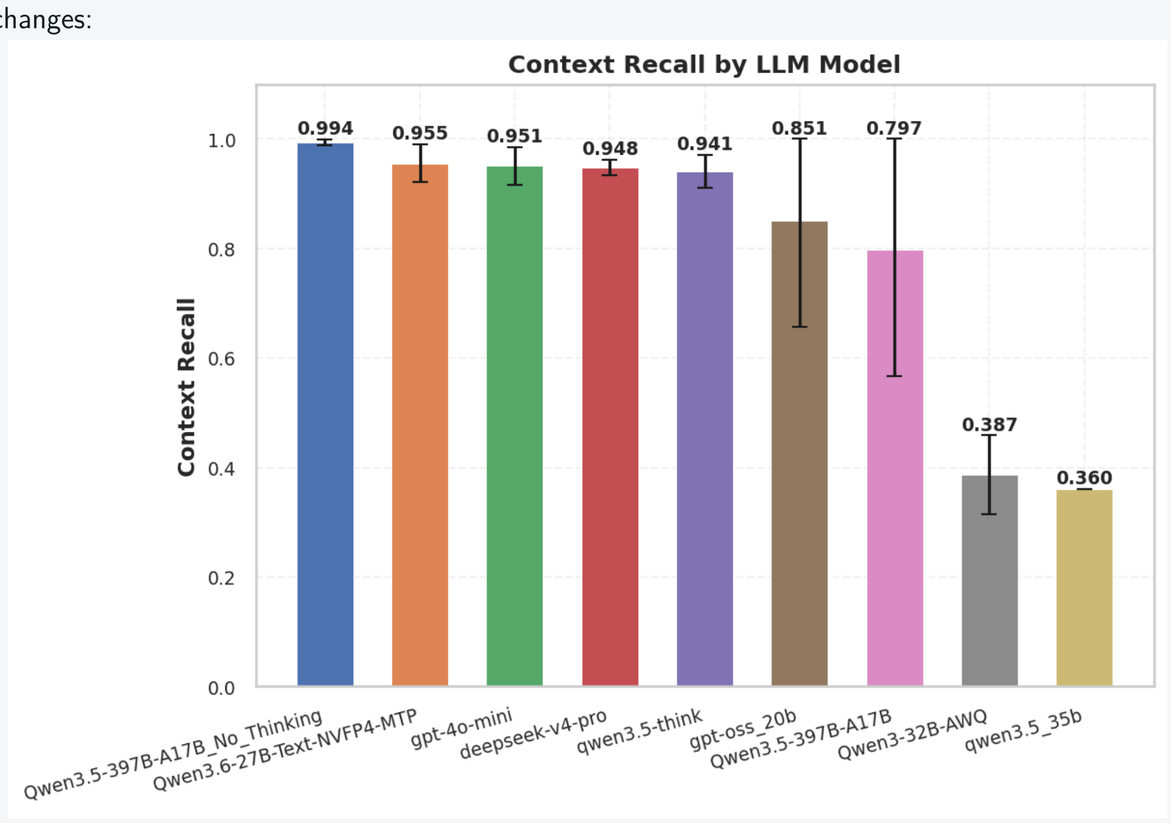
- **One grid, many systems:** swap chunker, retriever, prompt, reranker, model, or HTTP endpoint, no pipeline code changes.
- **Reproducible evidence:** every run writes metrics, timings, plots, traces, and MLflow under a unique run id.
- **Cross-run analysis:** plot scripts aggregate runs into the trade-off and heatmap figures shown on the right.

Benchmark Setup — Swept Variables

Variable	Values tested
Chunking	sizes $\{500, 1000\}$, overlaps $\{100, 200\}$, recursive / semantic
Retrieval	similarity, MMR; k=12
Prompt	concise, detailed
Generators	Qwen3-32B, Qwen3.5-think, Qwen3.6-27B, GPT-OSS-20B, GPT-4o-mini, DeepSeek-V4-Pro
Embedding	nomic-embed-text
Datasets	SQuAD, WikiQA (N=100 per config)

Retrieval Quality

Mean context recall per generator across the swept configurations. The ranking separates generators that consistently surface the gold context from those that lose it under aggressive chunking or prompt changes:



Variance bars show run-to-run spread. A high mean with small spread flags generators whose retrieval is both strong and stable under configuration changes, the most desirable property for production use.

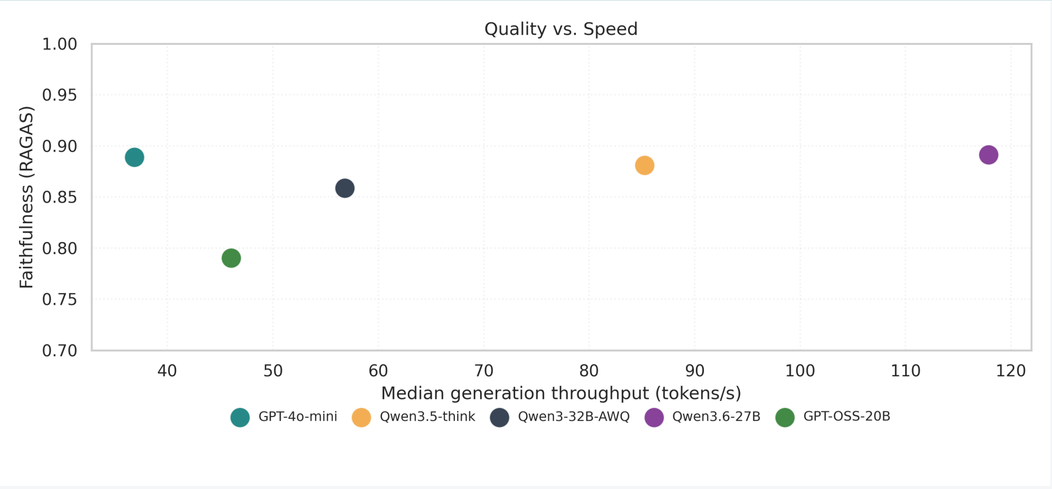
Key Findings

Three generators share top faithfulness at **0.89** (**GPT-4o-mini**, **Qwen3.6-27B**, **Qwen3.5-think** at **0.88**). Local **Qwen3.6-27B** matches GPT-4o-mini at a fraction of the inference cost, while **DeepSeek-V4-Pro** leads METEOR (**0.56**) but trails on retrieval. Critic model (Qwen3.5-397B-A17B) excluded from the generation sweep.

Generator	Faith. ↑	BERT-F1 ↑	nDCG@5 ↑	METEOR ↑
Qwen3.5-think	0.88 (0.95)	0.86 (0.93)	0.74 (0.81)	0.49 (0.60)
GPT-4o-mini	0.89 (0.96)	0.87 (0.93)	0.71 (0.80)	0.49 (0.59)
DeepSeek-V4-Pro	0.88 (0.94)	0.91 (0.92)	0.70 (0.73)	0.56 (0.66)
Qwen3.6-27B	0.89 (0.95)	0.89 (0.95)	0.70 (0.80)	0.51 (0.61)
Qwen3-32B-AWQ	0.83 (0.92)	0.91 (0.97)	0.77 (1.00)	0.48 (0.60)
GPT-OSS-20B	0.77 (0.84)	0.90 (0.96)	0.73 (0.80)	0.46 (0.60)

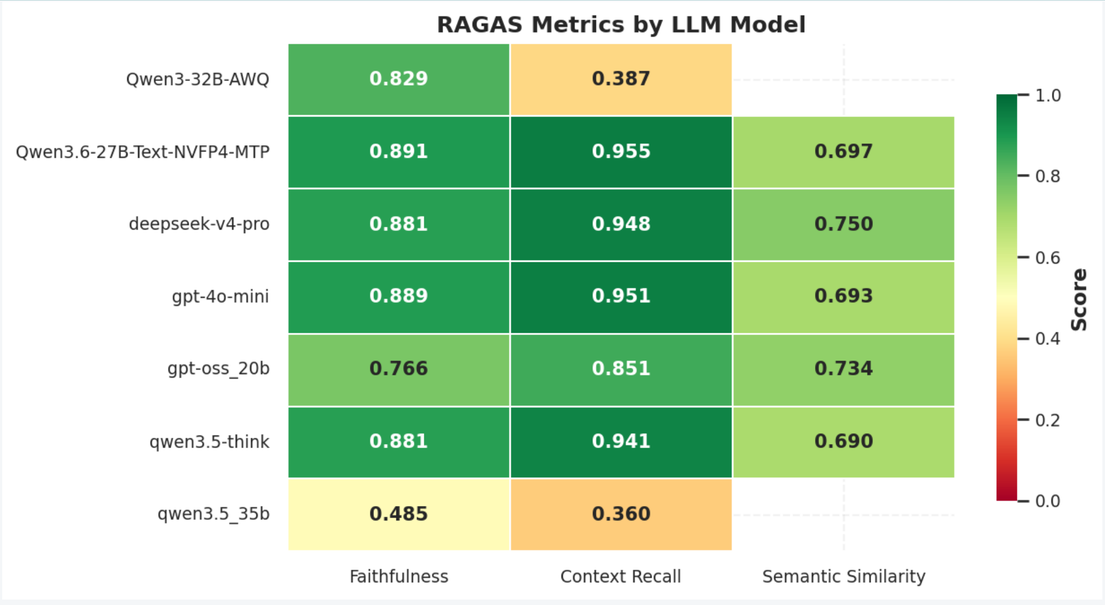
- **Prompting:** detailed prompts raise faithfulness by **+10 to +11 pp** on the larger generators (GPT-4o-mini, Qwen3.6-27B).
- **Retrieval:** MMR lifts nDCG@5 by **+4 to +11 pp**; faithfulness effect reverses with model size.

Quality vs. Speed



Faithfulness vs. median generation throughput. No single model dominates.

Prompt and Model Effects — Best Config per Model



Each cell: **best config per model** (picked by mean faithfulness), split by prompt template (concise vs. detailed). Prompt and model shift different quality metrics, so the framework reports multiple lenses.

Reproducibility Controls

- **Run evidence:** JSON/CSV, plots, MLflow, traces, timings in results/runN/.
- **Caching + audit:** fingerprints prevent re-embedding; artifacts keep later checks repeatable.

Limitations and Next Steps

- **Validity:** local hardware, stochastic generation, provider changes, dataset transfer.
- **Scale & ablations:** raise N, add significance tests; extend corpora, reranker sweeps, backend comparison.

Practical Takeaways

- **Complete configs:** benchmark whole pipelines, not isolated components.
- **Multi-metric:** each metric rewards different behavior.
- **Prompt + retrieval:** treat as first-class tuning variables.

Conclusion and References

RAG-NIMBF shows why complete configurations should be benchmarked, not isolated components. Prompt template, retrieval strategy, model choice, and metric choice can each change which configuration looks best.

1. Lewis et al. *RAG for Knowledge-Intensive NLP Tasks.* NeurIPS 2020.
2. Es et al. *RAGAS: Automated Evaluation of RAG.* arXiv:2309.15217, 2023.
3. Rajpurkar et al. *SQuAD: 100,000+ Questions for Machine Comprehension.* EMNLP 2016.

scan for code