Beyond Grid Search: Leveraging Bayesian Optimization for Accelerating RAG Pipeline Optimization

Anum Afzal
Technical University of Munich
anum.afzal@tum.de

Xueru Zheng
Technical University of Munich
xueru.zheng@tum.de

Florian Matthes
Technical University of Munich
matthes@tum.de

Abstract

Finding optimal configurations for Retrieval-Augmented Generation (RAG) pipelines via grid search is computationally prohibitive, limiting real-world scalability. We investigate Bayesian Optimization (BO) as an efficient alternative, systematically comparing seven BO strategies combining four surrogate models and two multi-fidelity methods across FiQA, SciFact, and HotpotQA datasets. Our framework explores both global pipeline and local component-wise optimization, targeting final RAG performance and resource efficiency. Our results show that BO reduces optimization time by up to 84% compared to grid search while maintaining comparable accuracy, with local optimization offering the most practical balance for deployment. Notably, performance gains plateau with larger evaluation budgets, suggesting that moderate resource investments suffice for effective RAG tuning. We provide actionable guidelines that empower industry practitioners to efficiently configure and deploy high-performing RAG systems under real-world constraints.

1 Introduction

Given its effectiveness in incorporating custom data, Retrieval Augmented Generation (RAG) has quickly found its place in Enterprise AI applications that require domain-specific knowledge to be injected into a Large Language Model (LLM). Achieving optimal performance through RAG requires careful configuration of multiple components, including retrievers, rerankers, filters, compressors, and generators. Given the vast array of optimization strategies and associated hyperparameters, finding optimal configurations through grid search becomes computationally prohibitive as the search space grows exponentially with each added component. This often hinders enterprises from easily scaling their RAG applications, especially with the constant surge of newly available models. AutoRAG (Kim et al., 2024) introduces automated configuration selection but relies on a grid search that cannot efficiently handle continuous parameters or leverage information from previous evaluations to guide the search process.

Bayesian Optimization (BO) is a global optimization technique that models the objective function using a probabilistic surrogate (typically Gaussian Processes or tree-based models) and selects configurations by balancing exploration and exploitation (Jones et al., 1998; Shahriari et al., 2016). BO has been widely used in hyperparameter optimization of Machine Learning models, and now recent work (Fu et al., 2024; Barker et al., 2025; Aravind, 2024; Conway et al., 2025) has focused on applying BO to RAG pipeline tuning. However, existing approaches typically optimize only limited hyperparameter subsets rather than the full configuration space, and no systematic comparison exists of different BO algorithms for RAG optimization. Furthermore, the trade-offs between global pipeline optimization versus local component-wise optimization remain unexplored, and how optimization effectiveness varies across different dataset domains is not well understood. We address these gaps by extending AutoRAG with a comprehensive BO framework that handles both discrete component selection and continuous hyperparameter optimization simultaneously. Our contributions are as follows:

- We present a BO-driven optimization framework that efficiently tunes the RAG component and hyperparameters, striking a balance between optimization time and RAG performance.
- We compare seven BO strategies combining four surrogate models (Random Forest, Tree-structured Parzen Estimator, Gaussian Process, and Heteroscedastic Gaussian Process) with two multi-fidelity methods (Successive

Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track), pages 263–277
March 25-27, 2026 ©2026 Association for Computational Linguistics

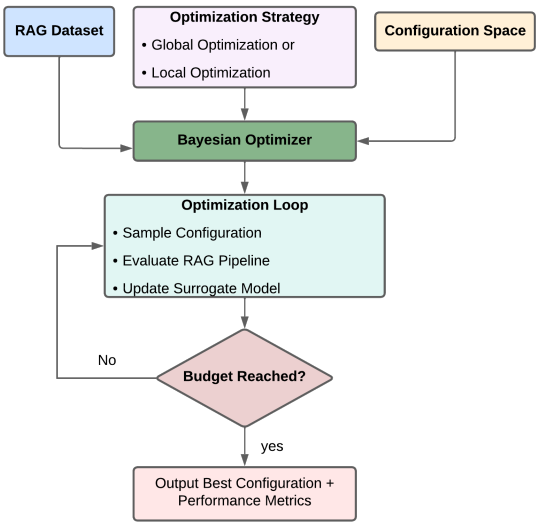


Figure 1: Overview of our Bayesian Optimization framework for RAG pipeline configuration. The framework supports both global optimization (entire pipeline) and local optimization (component-wise), using multiple surrogate models and multi-fidelity methods. Configuration Space includes both pipeline components and hyperparameters.

Halving and Hyperband) that terminate poorly performing trials early.

- We systematically investigate global optimization that treats the entire pipeline as a single objective versus local optimization that independently optimizes individual components.
- We evaluate these strategies across three datasets (FiQA, SciFact, and HotpotQA) representing different domains and complexity levels.

2 Related Work

AutoRAG (Kim et al., 2024) introduced the first end-to-end framework for automated RAG pipeline optimization, spanning data preprocessing, component selection, and evaluation through a node-based architecture. It uses local grid search to optimize each component (retriever, reranker, generator) independently, assuming individually optimal modules form an optimal pipeline. However, this approach ignores cross-component interactions and, due to its exhaustive and discretized grid search, fails to efficiently fine-tune continuous hyperparameters. Bayesian Optimization (BO) has proven highly effective for hyperparameter tuning in machine learning. Early systems such as Spearmint (Snoek et al., 2012), Auto-WEKA (Thornton et al., 2013), and Auto-sklearn (Feurer et al., 2015) demonstrated BO's superiority over grid and random search by effectively handling complex, hierarchical configuration space. Prior work has explored various applications of Bayesian Optimization in RAG pipeline tuning. AutoRAG-HP (Fu et al., 2024) formulates hyperparameter tuning as a multi-armed bandit problem, while recent work (Barker et al., 2025) introduces multi-objective optimization for cost and latency trade-offs. RAGBuilder (Aravind, 2024) focuses on continuous hyperparameter tuning, and Syftr (Conway et al., 2025) performs large-scale Bayesian optimization over agentic and non-agentic RAG pipelines to discover Pareto-optimal configurations balancing accuracy and cost. However, these approaches target limited subsets of hyperparameters and lack systematic comparisons of BO algorithms or optimization scopes, leaving open questions about the most effective strategies for comprehensive RAG pipeline optimization.

3 Methodology

3.1 RAG Pipeline

Our framework builds on AutoRAG's modular pipeline architecture and as shown in Figure 2 includes a RAG architecture of six main components: 1) Retrievers that find relevant documents from the corpus (Karpukhin et al., 2020; Robertson and Zaragoza, 2009), 2) Rerankers that refine retrieval results (Nogueira and Cho, 2020), 3) Filters that remove irrelevant content, 4) Compressors that reduce token usage (Xu et al., 2023), 5) Prompt makers that construct input prompts to guide downstream inference (Liu et al., 2021) and lastly 6) Generators that produce final responses (Raffel et al., 2023; Radford and Narasimhan, 2018; Ouyang et al., 2022). Additional details regarding the individual pipeline components can be in Appendix A.

3.2 Bayesian Optimization Framework

Figure 1 provides an overview of our proposed framework, illustrating how the RAG pipeline integrates with the Bayesian Optimization process. The diagram summarizes the end-to-end workflow, from input data and configuration space definition to the optimization loop and final selection of Pareto-optimal configurations. Bayesian Optimization provides an efficient approach to navigate

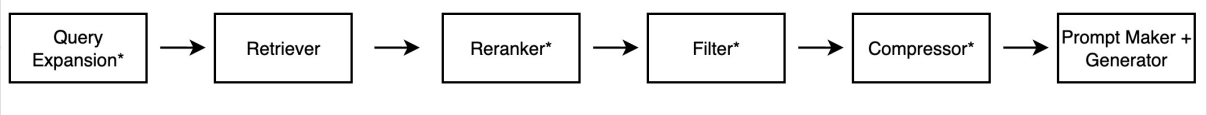


Figure 2: Sequential RAG pipeline architecture with six processing stages from query to response generation. Components marked with asterisks support pass-through functionality, enabling the optimizer to bypass stages when they do not improve performance.

this complex configuration space by building probabilistic surrogate models of the objective function and using acquisition functions to balance exploration of uncertain regions against exploitation of known high-performing areas (Jones et al., 1998; Shahriari et al., 2016).

We implement seven BO strategies by combining four surrogate models with multi-fidelity methods (Kandasamy et al., 2017). These strategies leverage multiple optimization libraries and encompass a variety of surrogate models and multi-fidelity techniques. Our framework includes both traditional single-fidelity approaches and advanced multi-fidelity variants that enhance sample efficiency, alongside support for multi-objective optimization. Specifically, our multi-objective optimization jointly targets performance quality and resource efficiency. Unlike single-objective methods that solely maximize evaluation scores, this approach balances performance against computational cost, enabling practitioners to identify configurations that deliver strong results within limited budgets (Deb et al., 2002; Coello Coello, 2006). The optimization considers both final performance scores and the time required to reach optimal configurations, yielding Pareto-optimal solutions that reflect different trade-offs between these objectives.

Surrogate Models Random Forest (RF) models use ensembles of decision trees to handle mixed discrete-continuous spaces effectively (Breiman, 2001; Lindauer et al., 2022). Tree-structured Parzen Estimator (TPE) models the distribution of good and bad configurations separately using kernel density estimation (Bergstra et al., 2011; Akiba et al., 2019; Falkner et al., 2018). Gaussian Process (GP) provides a probabilistic framework with smooth predictions and well-calibrated uncertainty estimates (Rasmussen and Williams, 2005). Heteroscedastic Gaussian Process (HGP) extends GP by modeling input-dependent noise, which is particularly useful when evaluation variance changes across the configuration space (Cowen-Rivers et al., 2022).

Multi-fidelity Methods Multi-fidelity methods reduce computational cost by terminating poorly performing trials early rather than evaluating all configurations with the full budget. Successive Halving (Jamieson and Talwalkar, 2015) allocates equal initial budget to all configurations, then iteratively eliminates the worst half while doubling resources for survivors. Hyperband (Li et al., 2018) extends this by running multiple Successive Halving brackets with different resource allocation strategies, providing robustness across different optimization landscapes. These methods are particularly valuable for RAG optimization, where full pipeline evaluation on complete datasets is expensive.

3.3 Global vs Local Optimization

We investigate two fundamental optimization strategies that differ in how they decompose the configuration problem. Global optimization treats the entire RAG pipeline as a single black-box function, jointly optimizing all components and their hyperparameters in one search. This approach can capture interdependencies between components, such as how retriever settings influence reranker performance, but it must explore an exponentially large joint configuration space that quickly becomes computationally expensive.

Local optimization, in contrast, follows a sequential component-wise strategy. Each component is optimized independently using a fixed evaluation budget, and the best-performing configuration from one stage is passed forward to the next. This avoids re-evaluating previously optimized components while still allowing downstream modules to adapt to upstream choices. Although this decomposition cannot fully model cross-component interactions, it substantially reduces search complexity and runtime by focusing the optimization on smaller, more manageable subspaces. The trade-off between these strategies reflects a central question in pipeline optimization: whether the added expressiveness of global search justifies its greater computational cost.

3.4 Dataset

We evaluate our optimization framework across three datasets that span diverse domains and reasoning requirements. FiQA (Maia et al., 2018) focuses on financial question answering and requires domain-specific understanding of markets, investments, and economic concepts. SciFact (Wadden et al., 2020) contains scientific claims with supporting or refuting evidence from biomedical literature, demanding precise fact verification and technical comprehension. HotpotQA (Yang et al., 2018), used in our work via the BEIR benchmark suite (Thakur et al., 2021), presents multi-hop reasoning questions that require aggregating information from multiple documents.

3.5 Evaluation Metrics

We employ a comprehensive evaluation framework that assesses both retrieval and generation quality, tailored to the specific components of the RAG pipeline.

Retrieval For retrieval evaluation (retriever, reranker, and filter), we compute document-level F1 scores by comparing the top-k predicted document IDs against the ground-truth relevant IDs, measuring how accurately the system identifies essential evidence.

Compressor For compression, we initially used token-level F1 to compare the compressed content with reference passages, but found that it lacked sensitivity to semantic preservation and contextual relevance. As a result, we adopted LLM-based evaluation, using large language models as judges to assess the quality of compressed content in a more human-aligned manner (Liu et al., 2023; Zheng et al., 2023).

Generator For generation, we report the arithmetic mean of four complementary metrics: BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004), and semantic similarity (Aynetdinov and Akbik, 2024), capturing both surface overlap and deeper meaning alignment with reference answers.

RAG The final optimization objective combines retrieval and generation scores equally (50/50), ensuring balanced optimization across the pipeline. We additionally validate final configurations using RAGAS(Es et al., 2025), which provides an end-to-end RAG-specific evaluation framework. Specifically, we employ the Faithfulness, Answer Relevance, LLM Context Precision Without Reference, Context Recall, Factual Correctness, and Semantic Similarity metrics for comprehensive post-optimization validation.

4 Experimental Setup

Each optimization run is constrained to an evaluation budget of 50 to 200 trials, informed by preliminary experiments on sample efficiency. Each dataset's validation set consists of 200 randomly sampled queries, selected to balance statistical reliability with computational efficiency. The 200-query size ensures consistent evaluation across experiments while keeping optimization runs computationally feasible, with each 50-trial run processing roughly 10,000 total queries through the full RAG pipeline. Validation sets remain fixed across all experiments to ensure fair comparison between optimization strategies without variance from query selection.

4.1 Baseline

We use RAG pipeline optimization using a grid search as our baseline. However, global optimization is intractable due to the full pipeline's configuration space exceeding 50 million possible combinations. Since exhaustive search is only feasible in much smaller configuration spaces, we limit grid search baselines to local optimization, where component-level subsets can be fully explored. To approximate the upper-bound performance of global search, we adopt a sequential local grid search strategy: each component is optimized in isolation, and its best configuration is fixed and passed to the next.

4.2 Models

To ensure comprehensive evaluation, we test the framework under two model configurations: (1) API-based embedding, reranker, and generator models, representing enterprise-grade proprietary baselines, and (2) open-source models deployed locally using an Nvidia A100 GPU with compressor modules leveraging OpenAI models, representing a non-enterprise setting. This dual setup enables robustness testing across industrial and open-source deployment contexts. For detailed model specifications, including embedding, reranker, and generator models, please refer to Appendix B.

5 Results and Discussion

This section addresses key aspects of Bayesian Optimization (BO) for Retrieval-Augmented Generation (RAG) pipelines. Due to space considerations, we show complete RAGAS scores in Appendix F.

5.1 Comparative Analysis of Bayesian Optimization Algorithms

Table 1 presents the full results for the comparative analysis of Bayesian Optimization (BO) algorithms across the SciFact, FIQA, and HotpotQA datasets. Overall, the most consistent and robust performance was obtained from SMAC3 with a Random Forest surrogate and Optuna with the Tree-structured Parzen Estimator (TPE). Both methods significantly outperformed random search and other BO variants, achieving higher final scores within comparable runtimes. SMAC3 with Random Forest achieved the best overall results, while Optuna-TPE provided a strong alternative with slightly lower peak performance and stable behavior across datasets. In contrast, multi-fidelity approaches such as SMAC3 with Successive Halving or Hyperband and RayTune with TPE + Hyperband did not perform reliably in this setting. Although theoretically more efficient, these methods tended to eliminate promising configurations too early due to partial-budget evaluations, which failed to capture full configuration effectiveness under noisy RAG metrics. Overall, the results indicate that full-budget Bayesian optimization remains the most effective strategy for RAG pipeline tuning under constrained evaluation budgets. Consequently, subsequent analyses focus on SMAC3 and Optuna-TPE as representative optimizers. See Appendix C for additional experimental analysis.

Lib.[a]	Surrogate	MF	MO	Score	Time
SciFact					
SMAC3	RF+HB	✓	✓	0.5339	1h01m
SMAC3	RF+SH	✓	✓	0.3998	49m
SMAC3	RF	×	✓	**0.6759**	1h40m
Optuna	TPE	×	✓	0.6288	1h45m
Optuna	GP	×	✓	0.6278	1h22m
RayTune	TPE+HB	✓	×	0.3637	1h12m
HEBO	HGP	×	✓	0.6226	2h25m
Optuna	Rand.	-	-	0.5786	2h18m
FIQA					
SMAC3	RF+HB	✓	✓	0.3926	1h03m
SMAC3	RF+SH	✓	✓	0.3564	47m
SMAC3	RF	×	✓	**0.4976**	2h43m
Optuna	TPE	×	✓	0.4688	1h42m
Optuna	GP	×	✓	0.4168	2h22m
RayTune	TPE+HB	✓	×	0.3913	3h11m
HEBO	HGP	×	✓	0.4111	2h30m
Optuna	Rand.	-	-	0.3826	2h18m
HotpotQA					
SMAC3	RF+HB	✓	✓	0.7137	1h20m
SMAC3	RF+SH	✓	✓	0.7224	1h39m
SMAC3	RF	×	✓	**0.7441**	2h37m
Optuna	TPE	×	✓	0.7437	2h45m
Optuna	GP	×	✓	0.6716	1h52m
RayTune	TPE+HB	✓	×	0.7112	2h13m
HEBO	HGP	×	✓	0.7105	2h55m
Optuna	Rand.	-	-	0.7309	3h12m

[a] Libraries used: SMAC3 (Lindauer et al., 2022), Optuna (Akiba et al., 2019), HEBO (Cowen-Rivers et al., 2022), RayTune (Liaw et al., 2018).

Table 1: Comparison of Bayesian Optimization algorithms (50-trial budget, Locally deployed models). Best scores per dataset in bold. RF = Random Forest, HB = Hyperband, SH = Successive Halving, GP = Gaussian Process, HGP = Heteroscedastic GP, Rand. = Random Search baseline, MF = Multi-Fidelity, MO = Multi-Objective

5.2 Scope of Bayesian Optimization in RAG Pipelines

Table 2 summarizes the local and global optimization, compared against the local grid search baseline, which exhaustively evaluates all configurations within each component. Each local optimization run evaluated 20 configurations per component using a sequential search strategy, while global optimization explored 50 configurations across the entire pipeline due to its substantially larger configuration space. Across datasets, both SMAC3 and Optuna TPE achieved comparable or higher scores than the local grid search baseline while substantially reducing runtime. The most significant efficiency gain was observed on the SciFact dataset, where SMAC3 reduced total optimization time by approximately 84% relative to grid search while achieving near-equivalent performance. Global optimization exhibited higher variance and longer runtimes but occasionally achieved slightly higher scores than local grid search, particularly on SciFact. Overall, local optimization provided the best balance between computational efficiency and accuracy, confirming that sequential component-wise Bayesian Optimization is an effective strategy for RAG pipelines under realistic resource constraints. Detailed component-wise scores and complete results for API-based models are presented in Appendix D.

Method	Opt	Score	Total Time
SciFact			
Grid Search	Local	0.6159	8h36m
SMAC3	Local	0.5768	**1h22m**
Optuna TPE	Local	0.5914	3h16m
SMAC3	Global	**0.6313**	4h25m
Optuna TPE	Global	0.6265	4h02m
FIQA			
Grid Search	Local	0.4682	8h35m
SMAC3	Local	0.4212	4h45m
Optuna TPE	Local	0.4775	**2h30m**
SMAC3	Global	0.4464	9h09m
Optuna TPE	Global	**0.4868**	6h12m
HotpotQA			
Grid Search	Local	0.6961	5h05m
SMAC3	Local	**0.6975**	1h48m
Optuna TPE	Local	0.5873	**1h09m**
SMAC3	Global	0.5728	3h57m
Optuna TPE	Global	0.5944	2h10m

Table 2: Comparison of local and global optimization results across datasets (locally deployed models). Each method was evaluated with 20 (local) or 50 (global) trials. Combined scores and total optimization times are reported. Opt refers to the Optimization Type.

5.3 Effect of Sample Size on Optimization Efficiency

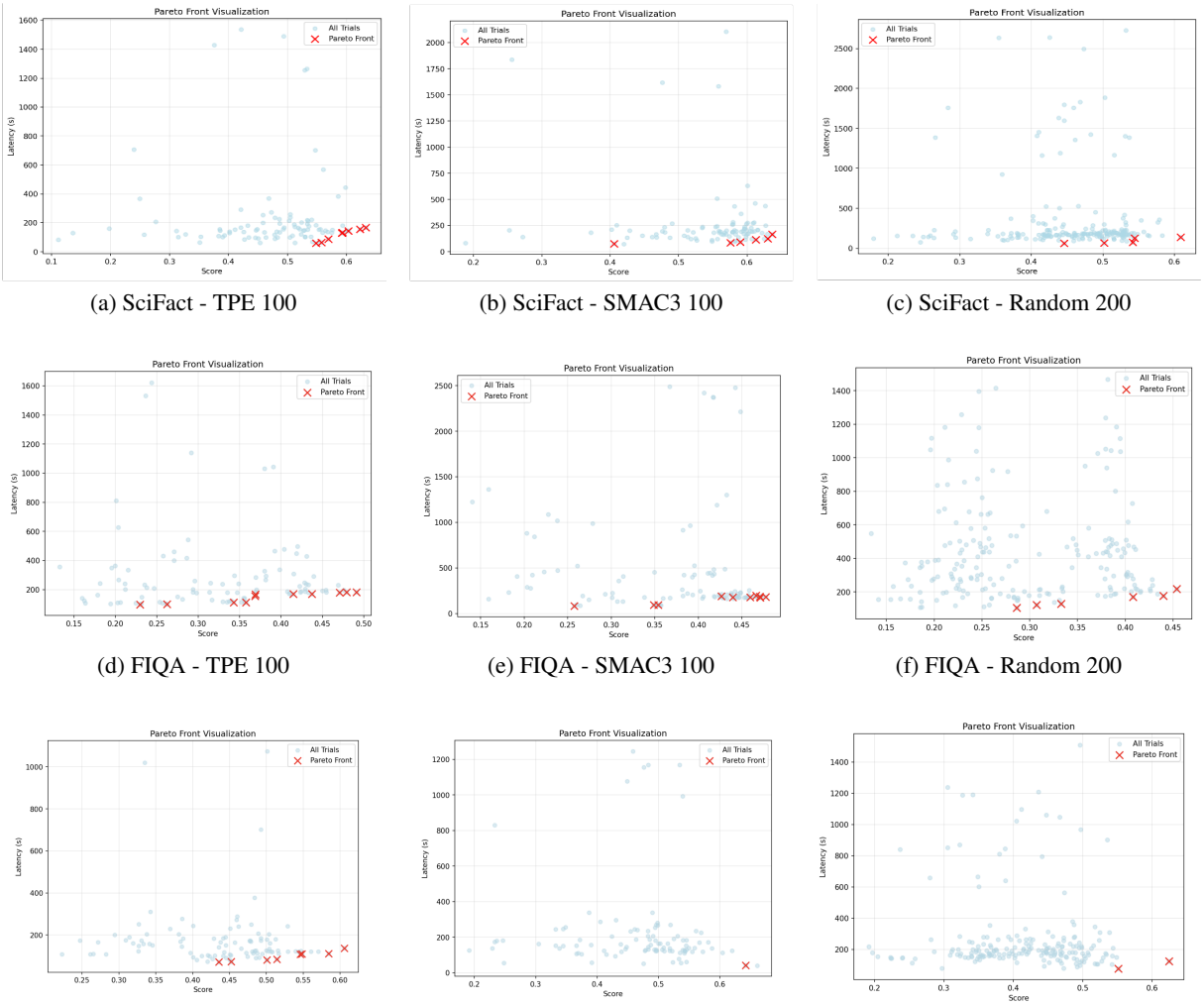
We evaluate if increasing the BO sampling budget from 50 to 100 configurations influences RAG tuning performance, with summary results shown in Table 3. The random search baseline samples configurations globally while respecting the pipeline's inter-component dependency constraints, providing a broad but inefficient exploration of the configuration space. Experimental Results outlined in Appendix E show that the impact of sample size varied notably across datasets. On SciFact, larger budgets yielded marginal improvements of less than 1%, indicating early convergence. FIQA showed moderate gains of 4–6%, reflecting the benefits of additional exploration in a sparse optimization landscape. In contrast, HotpotQA exhibited a substantial improvement of over 15% with SMAC3, suggesting that complex multi-hop reasoning tasks profit from larger search budgets. Interestingly, doubling the number of BO trials did not proportionally increase total optimization time. This sublinear runtime growth indicates that BO becomes more sample-efficient as the search progresses, concentrating evaluations on promising regions of the configuration space during later iterations. Despite the varying gains, both SMAC3 and Optuna TPE consistently outperformed the random-200 baseline, achieving higher scores in less than half the runtime. These results highlight that while larger budgets can improve performance, BO's adaptive sampling enables strong results even under limited evaluation budgets.

Method	Trials	Best Score	Time
Local Grid Search	–	0.4682	8h35m
Global Random	200	0.4536	23h53m
SMAC3	50	0.4464	9h08m
SMAC3	100	0.4744	13h14m
TPE	50	0.4686	6h12m
TPE	100	**0.4910**	7h52m

Table 3: Effect of sample size on global optimization efficiency for FIQA.

5.4 Dataset Characteristics and Optimization Robustness

Our analysis shows that BO performance strongly depends on the underlying characteristics of individual dataset's optimization landscape. SciFact provides a smooth and well-structured optimization landscape with clear performance gradients and limited interaction between components. These properties allow both SMAC3 and TPE to consistently discover clusters of high-scoring, low-latency configurations. As visible in Fig. 3a–3c, BO methods produce dense Pareto fronts near the upper score range, whereas Random-200 yields a scattered set of trials with only a few competitive points. FIQA exhibits a sparse and harsh landscape in which most configurations score poorly, creating a narrow band of viable solutions. This sparsity makes random search highly unreliable, as seen in Fig. 3f, where only a few Pareto-optimal configurations emerge. In contrast, TPE and SMAC3 (Fig. 3d–3e) efficiently identify and exploit the small region of acceptable performance, forming compact Pareto fronts around the 0.45–0.47 range. HotpotQA presents a deceptive plateau where many configurations achieve moderate scores, but few approach the global optimum. This landscape, combined with strong inter-component dependencies, causes BO optimizers to converge prematurely, producing more diffuse and lower-quality Pareto fronts (Fig. 3g–3h). Random search performs even worse, revealing only a couple of valid trade-offs (Fig. 3i),



(a) SciFact - TPE 100 (b) SciFact - SMAC3 100 (c) SciFact - Random 200

(d) FIQA - TPE 100 (e) FIQA - SMAC3 100 (f) FIQA - Random 200

(g) HotpotQA - TPE 100 (h) HotpotQA - SMAC3 100 (i) HotpotQA - Random 200

Figure 3: Pareto front visualizations across all datasets comparing TPE-100, SMAC3-100, and Random-200 optimization methods. Red crosses indicate Pareto-optimal configurations balancing score and latency. Rows represent different datasets (SciFact, FIQA, HotpotQA) while columns represent different optimization methods (TPE, SMAC3, Random).

further highlighting the difficulty of locating optimal configurations in this domain. In summary, Bayesian Optimization thrives in domains with clear performance gradients or sparse success regions but struggles in deceptive, plateaued landscapes with strong component coupling.

6 Conclusion and Future Work

We evaluated Bayesian Optimization strategies for tuning Retrieval-Augmented Generation (RAG) pipelines across diverse datasets and model configurations. SMAC3 and Optuna TPE consistently achieved competitive performance under varying conditions, both outperforming random and grid search baselines. Across experiments, local optimization proved more practical than global optimization, maintaining near-baseline accuracy while reducing total runtime by up to 84%. Increasing the sampling budget from 50 to 100 configurations yielded only modest improvements, with gains largely dependent on dataset characteristics. Our results show that moderate evaluation budgets are sufficient for effective RAG optimization, as larger sample sizes yield diminishing returns and higher computational costs. These findings offer actionable guidance for industry practitioners seeking to efficiently configure and deploy high-quality RAG systems at scale, especially under realistic resource constraints. We see further automation, dynamic adaptation, and multilingual extensions as promising avenues for industry-ready NLP deployments.

7 Limitations

This work systematically evaluates Bayesian Optimization strategies for tuning RAG pipelines, comparing multiple algorithms, optimization scopes, and sampling budgets across diverse datasets and model configurations. However, several important limitations constrain the scope and generalizability of these findings. The first limitation concerns the evaluation process, which relies on language-model-based scoring rather than human-annotated ground truth. While this approach enables scalable experimentation, it may introduce systematic biases and reduce the interpretability of the reported performance differences.

A second limitation arises from computational resource constraints, which restricted the number of optimization trials and the sampling depth within the vast configuration space. These constraints may have prevented the discovery of globally optimal configurations and limited the analysis of scalability under larger budgets. Finally, the experiments assume static data distributions and fixed pipeline architectures, whereas real-world RAG systems typically operate under dynamic, multilingual, and continuously evolving conditions. Collectively, these factors indicate that while the findings provide meaningful insights into optimization behavior, further validation is necessary for large-scale and adaptive production deployments.

8 Acknowledgement

This research is supported by SAP@TUM Collaboration Lab fostering a research partnership between the Technical University of Munich and SAP SE. The authors would like to specially thank Atreya Biswas (SAP) for initiating this project and for their ongoing guidance and support.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. *Preprint*, arXiv:1907.10902.

Ashwin Aravind. 2024. Ragbuilder: Open source tool kit for rag hyperparameter tuning. `https://github.com/ragbuilder/ragbuilder`. Accessed: 2025-09-23.

Ansar Aynetdinov and Alan Akbik. 2024. Semscore: Automated evaluation of instruction-tuned llms based on semantic textual similarity. *Preprint*, arXiv:2401.17072.

Satanjeev Banerjee and Alon Lavie. 2005. 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*, pages 65–72.

Matthew Barker, Andrew Bell, Evan Thomas, James Carr, Thomas Andrews, and Umang Bhatt. 2025. Faster, cheaper, better: Multi-objective hyperparameter optimization for llm and rag systems. *Preprint*, arXiv:2502.18635.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Proceedings of the 25th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA. Curran Associates Inc.

Leo Breiman. 2001. Random forests. 45(1):5–32.

C.A. Coello Coello. 2006. Evolutionary multi-objective optimization: a historical view of the field. *IEEE Computational Intelligence Magazine*, 1(1):28–36.

Alexander Conway, Debadeepta Dey, Stefan Hackmann, Matthew Hausknecht, Michael Schmidt, Mark Steadman, and Nick Volynets. 2025. syftr: Pareto-optimal generative ai. *Preprint*, arXiv:2505.20266.

Alexander I. Cowen-Rivers, Wenlong Lyu, Rasul Tutunov, Zhi Wang, Antoine Grosnit, Ryan Rhys Griffiths, Alexandre Max Maraval, Hao Jianye, Jun Wang, Jan Peters, and Haitham Bou Ammar. 2022. Hebo pushing the limits of sample-efficient hyperparameter optimisation. *Preprint*, arXiv:2012.03826.

K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. 2002. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197.

Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. 2025. Ragas: Automated evaluation of retrieval augmented generation. *Preprint*, arXiv:2309.15217.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. *Preprint*, arXiv:1807.01774.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Tobias Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Proceedings of the 29th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, page 2755–2763. MIT Press.

Jia Fu, Xiaoting Qin, Fangkai Yang, Lu Wang, Jue Zhang, Qingwei Lin, Yubo Chen, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024. Autorag-hp: Automatic online hyper-parameter tuning for retrieval-augmented generation. *Preprint*, arXiv:2406.19251.

Kevin Jamieson and Ameet Talwalkar. 2015. Non-stochastic best arm identification and hyperparameter optimization. *Preprint*, arXiv:1502.07943.

Donald R Jones, Matthias Schonlau, and William J Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492.

Kirthevasan Kandasamy, Gautam Dasarathy, Jeff Schneider, and Barnabas Poczos. 2017. Multi-fidelity bayesian optimisation with continuous approximations. *Preprint*, arXiv:1703.06240.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. 2020. Dense passage retrieval for open-domain question answering. *Preprint*, arXiv:2004.04906.

Dongkyu Kim, Byoungwook Kim, Donggeon Han, and Matouš Eibich. 2024. Autorag: Automated framework for optimization of retrieval augmented generation pipeline. *Preprint*, arXiv:2410.20878.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Preprint*, arXiv:1603.06560.

Richard Liaw, Eric Liang, Robert Nishihara, Philipp Moritz, Joseph E. Gonzalez, and Ion Stoica. 2018. Tune: A research platform for distributed model selection and training. *Preprint*, arXiv:1807.05118.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhopf, René Sass, and Frank Hutter. 2022. Smac3: A versatile bayesian optimization package for hyperparameter optimization. *Preprint*, arXiv:2109.09831.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *Preprint*, arXiv:2107.13586.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. G-eval: Nlg evaluation using gpt-4 with better human alignment. *Preprint*, arXiv:2303.16634.

Macedo Maia, André Freitas, Alexandra Balahur, Siegfried Handschuh, Manel Zarrouk, and Brian Davis. 2018. Fiqa – financial opinion mining and question answering (fiqa-2018 challenge). `https://sites.google.com/view/fiqa/`.

Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage re-ranking with bert. *Preprint*, arXiv:1901.04085.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. *Preprint*, arXiv:2203.02155.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Carl Rasmussen and Christopher Williams. 2005. *Gaussian Processes for Machine Learning*.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P. Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. *Preprint*, arXiv:1206.2944.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

Chris Thornton, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. 2013. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. *Preprint*, arXiv:1208.3719.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online. Association for Computational Linguistics.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2023. Recomp: Improving retrieval-augmented lms with compression and selective augmentation. *Preprint*, arXiv:2310.04408.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *Preprint*, arXiv:1809.09600.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

A RAG Pipeline Details

Each component offers multiple implementation choices with associated hyperparameters. For example, retrievers must select both the retrieval algorithm (*dense*, or *sparse/BM25*) and tune parameters such as `top_k`. Rerankers similarly choose a model type (e.g., *cross-encoder*) and set parameters such as `top_k` or relevance thresholds. Filters specify a filtering method (e.g., *percentile*, *threshold*, or *semantic similarity*) and corresponding hyperparameters such as a percentile or score cutoff. Compressors define the compression technique (e.g., *LexRank*, *spaCy-based*) with hyperparameters such as `compression_ratio`, `threshold`, `damping`, and `max_iterations`, as well as the choice of underlying linguistic model used for sentence representation. Prompt makers employ various prompt templates to formulate the final query for the LLM, while generators adjust inference parameters such as `temperature` and `max_tokens`.

The overall configuration space thus combines categorical decisions (e.g., component types and algorithm selections), continuous hyperparameters (e.g., temperature, threshold values), and integer-valued parameters (e.g., `top_k`), resulting in a high-dimensional, mixed-type space that is challenging to optimize exhaustively.

B Model Specifications

Locally deployed embedding, reranker, and generator models are listed in Tables 4, 5, and 6, respectively. API-based embedding models and API-based reranker/generator models are shown in Tables 7 and 8. Together, these tables summarize all models used in our experiments across both open-source and enterprise API settings.

C Additional Results: Comparative Analysis of Bayesian Optimization Algorithms

All optimizers were evaluated under an identical budget of 50 trials per dataset, using locally deployed models. Performance comparison focused on best score achieved and optimization efficiency. Consistent with the main results, SMAC3 with a Random Forest surrogate achieved the highest overall scores, while Optuna with the Tree-structured Parzen Estimator (TPE) provided a strong, stable alternative. Multi-fidelity methods such as Hyperband and Successive Halving demonstrated shorter runtimes but lower overall performance, confirming their reduced effectiveness in the RAG optimization setting.

D Detailed Results: Local and Global Optimization

Tables 9–12 provide detailed component-wise and combined results for both local and global optimization experiments across all datasets. A "/" symbol indicates that a component was skipped due to dependency constraints within the pipeline. Specifically, the reranker's top-k parameter must not exceed the number of documents returned by the retriever, and when the reranker's top-k equals 1, the filter is omitted. These constraints ensure valid and consistent configuration combinations across all optimization runs.

Model	Checkpoint
BGE Small	huggingface_baai_bge_small
RuBERT	huggingface_cointegrated_rubert_tiny2
MPNet	huggingface_all_mpnet_base_v2
BGE-M3	huggingface_bge_m3

Table 4: Locally deployed Embedding Models

Module Type	Model / Checkpoint
monot5	castorini/monot5-base-msmarco-10k castorini/monot5-large-msmarco-10k unicamp-dl/ptt5-base-en-pt-msmarco-100k-v2 unicamp-dl/mt5-base-mmarco-v1
upr	–
colbert reranker	–
Cross-Encoder	cross-encoder/ms-marco-MiniLM-L12-v2 cross-encoder/ms-marco-TinyBERT-L2-v2 cross-encoder/stsb-distilroberta-base
BGE Reranker	BAAI/bge-reranker-large BAAI/bge-reranker-base
BGE LLM Reranker	BAAI/bge-reranker-v2-m3 BAAI/bge-reranker-v2-gemma
flashrank_reranker	ms-marco-MiniLM-L-12-v2 ms-marco-MultiBERT-L-12 rank-T5-flan

Table 5: Locally deployed Reranker Models by Module Type

Model	Checkpoint
Llama 2 7B Chat	meta-llama/Llama-2-7b-chat-hf
Llama 3.2 1B Instruct	meta-llama/Llama-3.2-1B-Instruct
Phi-3 Mini 4K	microsoft/Phi-3-mini-4k-instruct
Qwen 3 4B	Qwen/Qwen3-4B
Qwen 2.5 1.5B Instruct	Qwen/Qwen2.5-1.5B-Instruct
Gemma 2B	google/gemma-2b
Gemma 3 1B IT	google/gemma-3-1b-it
Gemma 2 2B IT	google/gemma-2-2b-it
DeepSeek R1 Distill Qwen 1.5B	deepseek-ai/DeepSeek-R1-Distill-Qwen-1.5B
Llama 2 7B Chat AWQ	TheBloke/Llama-2-7B-Chat-AWQ
Llama 2 13B Chat AWQ	TheBloke/Llama-2-13B-chat-AWQ
CodeLlama 7B Instruct AWQ	TheBloke/CodeLlama-7B-Instruct-AWQ
TinyLlama 1.1B Chat	TinyLlama/TinyLlama-1.1B-Chat-v1.0

Table 6: Locally deployed Generator Models

Provider	Model
OpenAI	text-embedding-3-large
OpenAI	text-embedding-3-small
OpenAI	text-embedding-ada-002
Google	gemini

Table 7: API-based Embedding Models

Provider	Model
Cohere	cohere-rerank-v3.5

(a) Reranker Model

Provider	Model
Mistral	mistralai-large-instruct
OpenAI	gpt-3.5-turbo
Google	Gemini-2.0-flash
Anthropic	claude-4-sonnet

(b) Generator Models

Table 8: API-based Reranker and Generator Models

Method	Query Exp. + Retriever	Reranker	Filter	Compressor	Prompt + Generator	Combined	Time
SciFact Dataset							
SMAC3	0.6702	0.7545	/	0.8025	0.5404	0.6715	6h06m49s
Optuna TPE	0.7643	0.7987	/	0.8497	0.4943	0.6720	**5h53m49s**
Grid Search	0.7747	0.8353	/	0.8702	0.5048	**0.6875**	18h43m34s
FIQA Dataset							
SMAC3	0.3822	0.3822	/	0.7125	0.4373	0.5749	11h38m00s
Optuna TPE	0.4522	0.4537	/	0.7100	0.4172	0.5636	7h22m00s
Grid Search	0.4646	0.4777	/	0.7370	0.4245	**0.5807**	20h24m38s
HotpotQA Dataset							
SMAC3	0.7858	0.7858	0.7858	0.7147	0.7055	0.7107	5h49m42s
Optuna TPE	0.9050	/	/	/	0.8177	**0.8584**	**3h22m28s**
Grid Search	0.9050	0.9050	0.9050	0.8260	0.6761	0.7510	15h26m43s

Table 9: Local optimization results across datasets (API-based Models). Scores are reported per component, with the final combined score and total runtime shown in the last two columns.

Dataset	Method	Combined Score	Total Time Used
SciFact	SMAC3	0.6335	12h46m55s
	Optuna TPE	**0.6663**	13h42m36s
FIQA	SMAC3	0.5207	17h48m08s
	Optuna TPE	**0.5482**	17h02m12s
HotpotQA	SMAC3	0.7128	8h38m30s
	Optuna TPE	**0.7236**	10h16m01s

Table 10: Global optimization results across datasets (API-based Models).

Method	Query Exp. + Retriever	Reranker	Filter	Compressor	Prompt + Generator	Combined	Time
SciFact Dataset							
SMAC3	0.7430	0.8265	/	0.7288	0.4248	0.5768	**1h21m55s**
Optuna TPE	0.7806	0.8175	0.8262	0.7695	0.4133	0.5914	3h16m19s
Grid Search	0.7175	0.8137	/	0.7750	0.4568	**0.6159**	8h35m58s
FIQA Dataset							
SMAC3	0.1494	0.2236	/	0.4667	0.3757	0.4212	4h44m45s
Optuna TPE	0.3381	0.3537	0.3595	0.5647	0.3903	**0.4775**	**2h29m50s**
Grid Search	0.3525	0.3525	/	0.5450	0.3913	0.4682	8h35m03s
HotpotQA Dataset							
SMAC3	0.7950	0.7950	0.7950	0.7107	0.6843	**0.6975**	1h48m15s
Optuna TPE	0.7583	0.7583	0.7583	0.6570	0.5176	0.5873	**1h08m42s**
Grid Search	0.9200	0.9200	0.9200	0.8210	0.5712	0.6961	5h04m59s

Table 11: Local optimization results across datasets (Locally deployed Models). Scores are reported per component, with the final combined score and total runtime shown in the last two columns.

Dataset	Method	Combined Score	Total Time Used
SciFact	SMAC3	**0.6313**	4h24m54s
	Optuna TPE	0.6265	4h01m48s
FIQA	SMAC3	0.4464	9h08m50s
	Optuna TPE	**0.4868**	6h12m14s
HotpotQA	SMAC3	0.5728	3h56m44s
	Optuna TPE	**0.5944**	2h10m14s

Table 12: Global optimization results across datasets (Locally deployed Models).

E Detailed Results: Effect of Sample Size on Bayesian Optimization

This section reports the detailed results comparing different sampling budgets for Bayesian Optimization (BO) and random search across datasets. Each experiment evaluates the impact of increasing BO samples from 50 to 100 configurations, while the random search baseline uses 200 configurations. The "Top Configuration Distribution" column summarizes how many configurations fall within specific score ranges, illustrating the density of high-performing configurations.
Note. For the HotpotQA dataset, the best Optuna TPE score (trial 12) occurs unusually early due to stochastic evaluation noise. Variability in GPU execution, language model responses, and embedding API latencies can cause identical configurations to produce slightly different scores across runs, occasionally leading the optimizer to identify high-performing configurations earlier by chance rather than as a result of extended exploration.

F RAGAS Evaluation Framework and Metric Definitions

To complement the quantitative optimization analysis, we further evaluate the final RAG configurations using the RAGAS framework (Es et al., 2025). RAGAS provides an end-to-end evaluation method specifically designed for retrieval-augmented generation systems, measuring both retrieval effectiveness and answer quality in a unified manner. This framework ensures that improvements observed during optimization correspond to genuine gains in factual accuracy, contextual relevance, and faithfulness of generated outputs.

RAGAS defines several component-level metrics, each capturing a specific aspect of RAG performance:

- **Context Precision** — Measures the proportion of retrieved context passages that are relevant to the query, reflecting the retrieval component's precision.
- **Context Recall** — Evaluates how completely the retrieval step captures all relevant information needed to answer the query.
- **Answer Relevancy** — Assesses the degree to which the generated response directly addresses the query, given the retrieved evidence.
- **Faithfulness** — Quantifies whether the generated content remains grounded in the retrieved documents, identifying hallucinations or unsupported statements.
- **Factual Correctness** — Measures factual alignment between the generated answer and ground truth, indicating how accurately information is conveyed.
- **Semantic Similarity** — Captures the semantic overlap between the generated response and a reference answer, allowing evaluation beyond surface-level wording.
- **Retrieval Mean** and **Generation Mean** — Represent averaged retrieval- and generation-phase scores, providing a compact view of subsystem performance.
- **RAGAS Mean** — The overall composite score summarizing the end-to-end quality of the RAG pipeline across all evaluated dimensions.

These metrics jointly offer a comprehensive view of RAG performance, enabling fair comparison across optimization strategies, datasets, and model settings. The results in Tables 14–16 provide a detailed breakdown of retrieval and generation quality across SMAC3, Optuna TPE, Grid Search, and Random baselines. Each table reports both per metric scores (such as Context Precision and Faithfulness) and aggregated scores including Retrieval Mean, Generation Mean, and the overall RAGAS Mean. This structure allows a clear comparison of how different optimization strategies influence each stage of the RAG pipeline and the final answer quality.

Across datasets, Grid Search consistently achieves the highest overall RAGAS Mean, reflecting its exhaustive exploration of the configuration space. However, BO methods such as SMAC3 and TPE achieve competitive results while requiring far fewer evaluations. On FIQA (Table 14), TPE and Grid Search outperform other methods, with BO methods showing strong semantic similarity and solid retrieval accuracy. HotpotQA (Table 15) displays a similar pattern, where TPE performs comparably to Grid Search in generation quality despite the dataset's multi hop reasoning difficulty. For SciFact (Table 16), the structured nature of the dataset produces consistently high retrieval and generation scores across all optimization methods. The

Method	Best Score (Trial #)	Total Time Used	Top Configuration Distribution	Improved with More Samples?
			SciFact Dataset	
Grid Search Local	0.6159	8h 35m 58s	–	–
Random 200	0.6086 (trial 158)	19h 48m 12s	1 config ∼0.58, 2 configs ∼0.57, 2 configs ∼0.56	–
SMAC3 50	0.6313 (trial 48)	4h 25m 04s	2 configs ∼0.62, 5 configs ∼0.61, 5 configs ∼0.60	–
SMAC3 100	**0.6366** (trial 54)	7h 02m 37s	2 configs ∼0.63, 5 configs ∼0.62, 6 configs ∼0.61	Yes (+0.0053)
Optuna TPE 50	0.6265 (trial 32)	4h 01m 48s	2 configs ∼0.61, 2 configs ∼0.60, 2 configs ∼0.59	–
Optuna TPE 100	0.6325 (trial 98)	6h 30m 28s	1 config ∼0.62, 3 configs ∼0.60, 4 configs ∼0.59	Yes (+0.0060)
			FIQA Dataset	
Grid Search Local	0.4682	8h 35m 03s	–	–
Random 200	0.4536 (trial 143)	23h 53m 07s	1 config ∼0.45, 2 configs ∼0.43, 4 configs ∼0.42	–
SMAC3 50	0.4464 (trial 20)	9h 08m 00s	5 configs ∼0.44, 3 configs ∼0.43, 4 configs ∼0.42	–
SMAC3 110	0.4744 (trial 108)	13h 14m 00s	6 configs ∼0.47, 5 configs ∼0.46, 7 configs ∼0.45	Yes (+0.0280)
Optuna TPE 50	0.4686 (trial 22)	6h 12m 14s	2 configs ∼0.46, 2 configs ∼0.45, 2 configs ∼0.44	–
Optuna TPE 100	**0.4910** (trial 45)	7h 52m 13s	3 configs ∼0.47, 2 configs ∼0.46, 4 configs ∼0.45	Yes (+0.0224)
			HotpotQA Dataset	
Grid Search Local	0.6961	5h 04m 59s	–	–
Random 200	0.6242 (trial 110)	15h 08m 03s	1 config ∼0.58, 2 configs ∼0.57, 2 configs ∼0.56	–
SMAC3 50	0.5727 (trial 48)	3h 56m 44s	2 configs ∼0.54, 1 config ∼0.53, 1 config ∼0.52	–
SMAC3 100	**0.6606** (trial 92)	6h 17m 59s	1 config ∼0.64, 1 config ∼0.61, 1 config ∼0.60	Yes (+0.0879)
Optuna TPE 50	0.5944 (trial 28)	2h 10m 14s	1 config ∼0.57, 2 configs ∼0.56, 1 config ∼0.55	–
Optuna TPE 100	0.6057 (trial 12)	4h 51m 10s	1 config ∼0.58, 1 config ∼0.57, 1 config ∼0.56	Yes (+0.0113)

Table 13: Results comparing sample sizes in global optimization across datasets. The "Top Configuration Distribution" column shows the distribution of configurations across score ranges.

close clustering of RAGAS Means shows that SciFact offers a stable optimization landscape where multiple strategies can achieve strong end-to-end performance.

These results confirm that the improvements identified during optimization translate into measurable gains in retrieval accuracy, factual grounding, and answer quality, supporting the effectiveness of Bayesian Optimization for tuning RAG pipelines.

Method	RAGAS Mean	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Factual Correctness	Semantic Similarity	Retrieval Mean	Generation Mean
SMAC3 Local	0.7633	**0.9800**	0.6055	0.6824	0.9376	0.4536	0.9208	0.7927	0.7486
Optuna TPE Local	0.7680	0.9550	0.6684	0.6456	0.9143	0.4905	0.9343	0.8117	0.7462
Grid Search Local	**0.7814**	**0.9800**	**0.6878**	0.6336	0.9485	**0.5025**	**0.9359**	**0.8339**	**0.7551**
SMAC3 Global 50	0.7424	**0.9800**	0.4961	**0.7140**	0.9413	0.4058	0.9171	0.7380	0.7445
Optuna TPE Global 50	0.7710	0.9750	0.6425	0.6708	**0.9806**	0.4263	0.9306	0.8088	0.7521
Random 200	0.6884	0.9000	0.5624	0.6372	0.8474	0.3051	0.8785	0.7312	0.6670

Table 14: RAGAS Scores for FIQA: API-based Optimization Results. Best scores per metric are highlighted in bold.

Method	RAGAS Mean	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Factual Correctness	Semantic Similarity	Retrieval Mean	Generation Mean
SMAC3 Local	0.7571	**0.8800**	0.6075	0.5521	**0.9786**	0.5875	0.9369	0.7437	0.7638
Optuna TPE Local	0.8075	0.7625	0.8125	0.8691	0.8036	0.6435	**0.9536**	0.7875	**0.8175**
Grid Search Local	**0.8172**	0.8650	0.7767	0.8648	0.8018	**0.6609**	0.9342	**0.8208**	0.8154
SMAC3 Global 50	0.7398	0.6970	0.6030	0.8547	0.7114	0.6226	0.9503	0.6500	0.7847
Optuna TPE Global 50	0.8116	0.7694	**0.8467**	**0.8775**	0.7790	0.6449	0.9523	0.8081	0.8134
Random Global 200	0.7774	0.8275	0.7208	0.8773	0.7094	0.5927	0.9370	0.7742	0.7791

Table 15: RAGAS Scores for HotpotQA: API-based Optimization Results. Best scores per metric are highlighted in bold.

Method	RAGAS Mean	Context Precision	Context Recall	Answer Relevancy	Faithfulness	Factual Correctness	Semantic Similarity	Retrieval Mean	Generation Mean
SMAC3 Local	0.8050	0.9550	0.7282	0.7422	0.9653	0.4920	0.9475	0.8416	0.7868
Optuna TPE Local	0.8415	0.9600	0.7859	0.8062	0.9657	**0.5730**	0.9580	0.8730	0.8257
Grid Search Local	**0.8525**	**0.9900**	**0.8209**	**0.8262**	0.9702	0.5496	**0.9583**	**0.9055**	**0.8261**
SMAC3 Global 50	0.7947	0.8950	0.7732	0.7242	0.8565	0.5645	0.9548	0.8341	0.7750
Optuna TPE Global 50	0.8331	0.9700	0.7496	0.8189	**0.9722**	0.5285	0.9594	0.8598	0.8198
Random Global 200	0.8434	0.9800	0.7895	0.8230	0.9679	0.5419	0.9583	0.8847	0.8228

Table 16: RAGAS Scores for SciFact: API-based Optimization Results. Best scores per metric are highlighted in bold.