information

MDPI

*Article*

# Black-Box Hyperparameter Optimization for Financial RAG Retrieval: An Efficiency–Effectiveness Trade-Off Study

**Yangyang Jin, Xindi Wang and Qianli Dong ***

School of Economics and Management, Beijing Jiaotong University, Beijing 100044, China; 23241214@bjtu.edu.cn (Y.J.); xdwang@bjtu.edu.cn (X.W.)
* Correspondence: 25120540@bjtu.edu.cn

**Abstract**

This study examines black-box hyperparameter optimization for financial retrieval-augmented generation (RAG) retrieval under limited budget constraints. Using FinQA as the primary dataset, it compares Grid Search, Random Search, and Bayesian Optimization under a unified search space, evaluation protocol, and multi-seed setting, and further uses FinanceBench for external validation. The results show that Random Search and Bayesian Optimization can approach the Grid reference at substantially lower cost, but the small development-set advantage of Bayesian Optimization does not remain stable on the test set or across repeated runs. A more consistent finding is that high-performing configurations are concentrated in a limited parameter region. Overall, the results suggest that, in budget-constrained financial RAG retrieval tuning, identifying stable high-performing parameter regions may be more useful than relying on increasingly complex optimization methods.

**Keywords:** financial RAG retrieval; black-box hyperparameter optimization; efficiency–effectiveness trade-off; Random Search; Bayesian Optimization; stable high-performance regions

## 1. Introduction

Retrieval-augmented generation (RAG) has become a common framework for domain-specific question answering [1,2]. This is particularly relevant in the financial domain, where documents are often long, structurally complex, and rich in dispersed evidence across narrative text, tables, and notes [3–6]. In such settings, retrieval quality directly affects the usefulness of the final answer, making retrieval design and tuning an important issue in financial RAG systems.

Existing studies have mainly focused on retrieval architectures, representation models, and ranking strategies [7–10], while giving relatively less attention to retrieval hyperparameter optimization [11]. In practice, parameters such as chunk size, chunk overlap, and fusion weight can affect text segmentation, contextual redundancy, and ranking behavior, and may therefore influence retrieval performance. Because these effects are observed only through full system evaluation, financial RAG retrieval tuning can be formulated as a black-box optimization problem [12,13]. Although Grid Search, Random Search, and Bayesian Optimization are widely used approaches for such problems [14], their relative behavior under limited budgets remains insufficiently examined in financial RAG retrieval, especially from the perspective of the efficiency–effectiveness trade-off.

This study therefore focuses on three related questions: whether different black-box optimization methods can approach high-quality configurations under limited budgets in

Check for updates

Academic Editor: Tieling Zhang

Received: 12 March 2026
Revised: 13 April 2026
Accepted: 15 April 2026
Published: 24 April 2026

**Copyright:** © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

financial RAG retrieval; whether Random Search and Bayesian Optimization provide a more favorable efficiency–effectiveness trade-off than exhaustive Grid Search; and whether strong retrieval performance is associated with relatively stable hyperparameter regions rather than with one specific optimization method. To investigate these questions, this study develops a comparative framework for black-box hyperparameter optimization in financial RAG retrieval and evaluates Grid Search, Random Search, and Bayesian Optimization under a unified setting. The main contributions are as follows:

1. It develops a comparative framework for black-box hyperparameter optimization in financial RAG retrieval and examines three optimization methods under limited budgets.
2. It provides an empirical comparison between low-budget optimization methods and exhaustive search from the perspective of the efficiency–effectiveness trade-off.
3. By combining parameter distribution analysis with multi-seed results, it examines the stability of high-performing configurations and summarizes representative high-performance parameter patterns.

## 2. Methodology and Experimental Design

### *2.1. Dataset and Experimental Setup*

This study uses two financial question-answering datasets. The FinQA dataset from the $T^2$-RAGBench serves as the benchmark for retrieval architecture comparison, hyperparameter optimization, and the main empirical analysis. It is derived from real-world financial reports and contains questions with substantial numerical reasoning requirements. Following the standard split, the development and test sets contain 883 and 1150 queries, respectively. In addition, the open-source subset of FinanceBench is used as an external validation dataset to examine whether the main findings remain consistent on a second financial QA benchmark. Since the publicly available version contains only 150 samples and does not provide a standard development–test split comparable to FinQA, it is used for external validation rather than as a fully equivalent primary optimization dataset.

Because the original data cannot be directly used for retrieval experiments, both datasets are converted into a retrieval-oriented format under a unified preprocessing principle. For FinQA, document-level contexts are separated into text and tables, after which the text is chunked to construct a chunk library and the corresponding query–chunk relevance labels. Figure 1 summarizes this workflow. For the FinanceBench open-source subset, questions are treated as retrieval queries, source documents are chunked in the same manner, and the provided evidence information is used to derive evidence-to-chunk mappings for evaluation. This procedure keeps the two datasets aligned in retrieval-unit definition, document reconstruction, and evaluation criteria.

Since automatically constructed query–chunk labels may be noisy, we manually validate the derived labels on both datasets. For FinQA, 536 query–chunk pairs are sampled from the development set. The results show 76.3% overall agreement and a Cohen's $\kappa$ of 0.66, indicating moderate agreement; detailed relevance rates by label source are reported in Table 1. For the FinanceBench open-source subset, we conduct a manual spot check of the evidence-to-chunk mappings. The results show 90.0% positive mapping accuracy, 85.0% negative mapping accuracy, 87.5% overall agreement, and a Cohen's $\kappa$ of 0.75; the corresponding results are reported in Table 2.

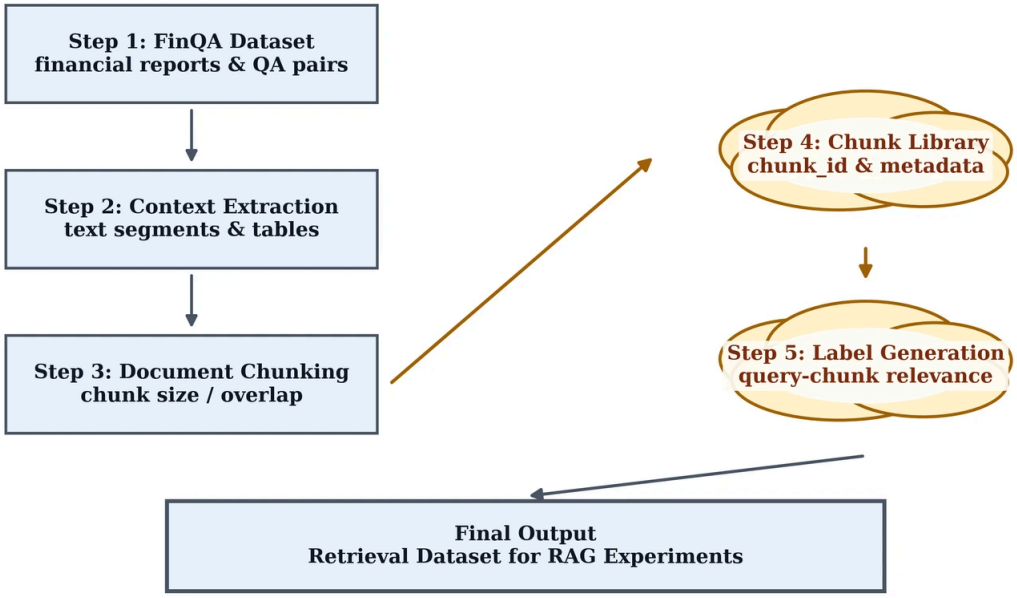



**Figure 1.** FinQA data processing pipeline for retrieval experiments.

**Table 1.** Manual relevance rates by label source on FinQA.

| Validation Item | Value |
|---|---|
| Answer match | 100.0% |
| Boundary negative | 57.6% |
| Random negative | 27.6% |
| Retrieved topk | 26.7% |
| Weak positive | 72.2% |

**Table 2.** Manual validation results for evidence-to-chunk mappings on the FinanceBench open-source subset.

| Validation Item | Value |
|---|---|
| Sampled pairs | 80 |
| Positive mapping accuracy | 90.0% |
| Negative mapping accuracy | 85.0% |
| Overall agreement | 87.5% |
| Cohen's κ | 0.75 |

To ensure comparability across retrieval architectures, optimization methods, and datasets, all experiments follow the same basic setup. All retrieval systems share the same preprocessing principles, chunking rules, evaluation metrics, and reporting format. In the primary experiments, the development set is used only for architecture selection and hyperparameter search, while the test set is reserved for final reporting and statistical inference. In the external validation stage, the FinanceBench open-source subset follows the same retrieval-oriented preprocessing principles and evaluation criteria as the FinQA-based main experiments. However, it is used only for external validation rather than as a fully equivalent benchmark for search-based hyperparameter optimization. All experiments were implemented in Python 3.13, and the retrieval, statistical analysis, and figure generation were conducted under the same saved-output evaluation workflow.

### 2.2. Retrieval Framework and Compared Architectures

Financial question-answering documents often contain specialized terminology, numerical information, and cross-sentence semantic dependencies [15]. As a result, retrieval performance is often influenced by both lexical and semantic signals. To systematically compare these signals and their combination, this study considers Dense Retrieval, BM25 Retrieval, and Hybrid Retrieval as the three retrieval architectures. They represent semantic matching, lexical matching, and their integration, respectively. Let $q$ denote the original query and $c$ denote a document chunk. For vector-based retrieval methods, both are first mapped into dense representations through an embedding function E(·):

$$e_q = E(q), \tag{1}$$

$$e_c = E(c). \tag{2}$$

Based on these representations, Dense Retrieval ranks document chunks according to semantic similarity between the query vector and the chunk vector. Its retrieval score is defined as [16]

$$S_{dense}(q,c) = sim(e_q, e_c), \tag{3}$$

where sim(·) denotes a similarity function such as cosine similarity.

BM25 Retrieval estimates document relevance based on lexical matching signals, and its score is written as [17]

$$S_{lex}(q,c) = BM25(q,c), \tag{4}$$

where $S_{lex}(q,c)$ denotes the lexical retrieval score between query $q$ and document chunk $c$.

To combine semantic and lexical signals, Hybrid Retrieval adopts a weighted fusion strategy [18]. Let $\widetilde{S}_{\text{dense}}(q,c)$ and $\widetilde{S}_{\text{lex}}(q,c)$ denote the normalized dense and lexical scores, respectively. The Hybrid Retrieval score is then defined as

$$S_{hyb}(q,c;\,\alpha) = \alpha\widetilde{S}_{\text{dense}}(q,c) + (1-\alpha)\widetilde{S}_{\text{lex}}(q,c), \tag{5}$$

where $\alpha \in [0,1]$ is the fusion weight controlling their relative contributions of the two signals.

For any retrieval architecture $r \in \{\text{dense}, \text{lex}, \text{hyb}\}$, the returned result set is denoted by

$$R_k^{(r)}(q) = TopK_{c\in C}(S_r(q,c), k), \tag{6}$$

where $\mathcal{C}$ is the set of document chunks and $R_k^{(r)}(q)$ represents the top-$k$ retrieval results returned for query $q$.

Under a unified experimental protocol, the three architectures are first compared in terms of retrieval performance, after which the architecture selected on the basis of the comparative results is used for subsequent hyperparameter optimization. To avoid relying solely on single-point estimates, the Results Section further reports paired statistical tests and effect size analysis for the difference between BM25 and Hybrid. To ensure fairness, all three architectures are evaluated on the same reconstructed document corpus, query set, and evaluation protocol. Dense Retrieval and Hybrid Retrieval share the same embedding model, while BM25 Retrieval is performed on the same chunk collection using lexical matching.

### *2.3. Hyperparameter Optimization Protocol*

#### 2.3.1. Problem Formulation and Objective

To examine how parameter configurations affect retrieval performance in financial RAG systems, retrieval tuning is formulated as a hyperparameter optimization problem on the selected base retriever. Let

$$x \in \mathcal{X}, \tag{7}$$

denote a hyperparameter configuration in the predefined search space $\mathcal{X}$, where the main variables considered in this study are chunk size, chunk overlap, and the fusion weight used in Hybrid Retrieval. For a given configuration $x$, the corresponding retrieval system is evaluated on the development set, and the optimization problem is defined as

$$x^* = arg \max_{x \in \chi} f(x) \tag{8}$$

where $f(x)$ denotes the retrieval objective of configuration $x$.

This problem is treated as a black-box optimization task because the objective can only be observed through full system evaluation, including document chunking, index construction, retrieval, and metric computation. Moreover, the mapping from hyperparameters to retrieval performance is generally non-differentiable in practice, as it depends on discrete chunk boundaries, candidate sets, and ranking outcomes. Accordingly, the focus of this study is not to develop a new optimizer, but to compare how different black-box search strategies behave under limited budgets.

The search space is defined as the Cartesian product of the candidate sets of the individual parameters:

$$\chi = \chi_{cs} \times \chi_{ov} \times \chi_{\alpha}, \tag{9}$$

where $\mathcal{X}_{cs}$, $\mathcal{X}_{ov}$, and $\mathcal{X}_{\alpha}$ denote the candidate sets for chunk size, chunk overlap, and fusion weight, respectively. To avoid optimizing only a single metric, hyperparameter search is guided by a composite objective:

$$f(x) = 0.4nDCG@5(x) + 0.3Recall@5(x) + 0.2MRR(x) + 0.1Precision@5(x). \tag{10}$$

This formulation balances ranking quality, coverage, and retrieval accuracy, and reduces the risk that the search process is dominated by a single evaluation metric.

#### 2.3.2. Optimization Methods

Three optimization strategies are compared: Grid Search, Random Search, and Bayesian Optimization.

Grid Search exhaustively enumerates all valid configurations in the predefined discrete search space and therefore serves as an exhaustive reference for the best achievable result within that space. Random Search samples configurations from $\mathcal{X}$ under a fixed trial budget and returns the best observed configuration. Bayesian Optimization selects the next configuration sequentially based on historical observations. In this study, it is implemented in a TPE-style manner, using past trials to guide subsequent proposals toward potentially high-performing regions [19]. Compared with Random Search, this allows more informed search under the same budget constraint.

These three methods are chosen because they represent exhaustive search, unguided sampling, and sequentially guided search, respectively. Their comparison makes it possible to assess how different search strategies behave under the current retrieval-tuning setting.

#### 2.3.3. Budget Setting and Multi-Seed Design

To compare efficiency–effectiveness trade-offs under the same search space, different budgets are assigned to the three methods. Grid Search evaluates all 60 valid configurations in the predefined space, whereas Random Search and Bayesian Optimization are each restricted to 20 trials per run. Thus, the latter two methods are compared under a clearly defined low-budget setting:

$$B \ll |X|. \tag{11}$$

This design is motivated by two considerations. First, each trial requires full retrieval evaluation and is therefore computationally expensive. Second, the objective is not only to identify the single best configuration, but also to examine whether low-budget black-box methods can approach the grid-based reference optimum at a substantially lower cost.

Because Random Search and Bayesian Optimization involve stochastic sampling, both methods are repeated under five random seeds (7, 42, 123, 2024, and 3407). For each run, the best configuration, runtime to best, and corresponding test-set performances are recorded. This design supports a fair comparison of search efficiency, result stability, and generalization performance under the same constraints.

The hyperparameter optimization protocol described above is applied to the primary benchmark, FinQA. For the external validation dataset, the FinanceBench open-source subset is not used for full search-based optimization; instead, it is used to examine whether the architecture-level findings and representative high-performing configurations identified on FinQA remain consistent across datasets. To further assess whether high performance is associated with reproducible parameter regions rather than isolated single-point optima, subsequent analysis also considers parameter interactions and the concentration of high-performing configurations.

### *2.4. Evaluation and Statistical Analysis*

#### 2.4.1. Evaluation Metrics

Retrieval performance is evaluated using Precision@5, Recall@5, MRR, and nDCG@5 [20,21]. During hyperparameter search, candidate configurations are selected according to the composite objective defined in Section 2.3.1, whereas final results are reported separately for all metrics.

For statistical inference, nDCG@5 is treated as the primary metric because it reflects both relevance and ranking quality. Recall@5 is used as a secondary metric for retrieval coverage. MRR and Precision@5 are reported as complementary descriptive indicators.

#### 2.4.2. Statistical Inference and Effect Size Reporting

Statistical inference is conducted only on the held-out test set, rather than on the development set used for architecture selection and hyperparameter search. Pairwise comparisons are performed on query-level nDCG@5 using a paired permutation test, and 95% bootstrap confidence intervals are computed for mean paired differences. When multiple pairwise comparisons are carried out within the same analysis, Holm correction is applied.

This framework is used for both architecture-level and configuration-level comparisons. The former focuses in particular on the difference between BM25 and Hybrid Retrieval, while the latter examines representative high-performing configurations identified under different optimization methods. For each comparison, the reported statistics include the mean paired difference, the corresponding 95% confidence interval, and the adjusted $p$-value. Effect size is interpreted primarily through the magnitude and interval estimate of the paired difference rather than through relative percentage improvement alone.

2.4.3. Result Reporting and Robustness

When repeated runs are involved, point estimates are not reported in isolation. For stochastic methods such as Random Search and Bayesian Optimization, results are summarized across five random seeds, together with their corresponding variability. For deterministic results, point estimates are supplemented with inferential comparisons on the held-out test set where applicable.

This reporting strategy reduces dependence on individual runs and helps align inferential claims with the available evidence. Accordingly, claims of superiority are made only when supported by paired statistical testing and uncertainty estimates, whereas the remaining metrics are treated as descriptive evidence.

## 3. Results

### *3.1. Retrieval Architecture Comparison*

We evaluate Dense, BM25, and Hybrid Retrieval on the FinQA dev set to compare the effectiveness of different retrieval architectures. The results are summarized in Table 3 and Figure 2.

**Table 3.** Different architecture results.

| Architecture | Recall@5 | Precision@5 | nDCG@5 | Recall@10 | Precision@10 | nDCG@10 | MRR |
|---|---|---|---|---|---|---|---|
| Dense | 0.5119 | 0.1574 | 0.3738 | 0.6474 | 0.1061 | 0.4271 | 0.4104 |
| BM25 | 0.5812 | 0.1796 | 0.4206 | 0.7069 | 0.1185 | 0.4721 | 0.4466 |
| Hybrid | 0.6012 | 0.1844 | 0.4386 | 0.7342 | 0.1229 | 0.4937 | 0.4769 |

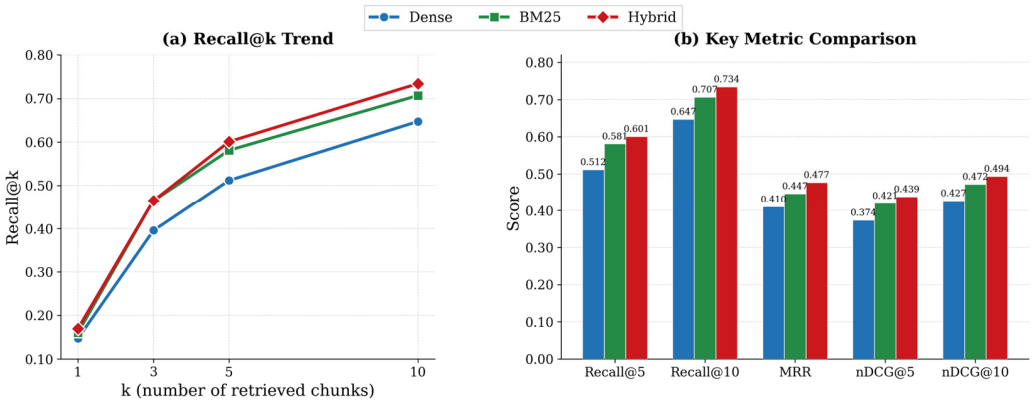



**Figure 2.** Retrieval architecture comparison on the FinQA dev set: (**a**) Recall@k trends of Dense, BM25, and Hybrid Retrieval; (**b**) grouped comparison of key retrieval metrics.

As shown in Figure 2, Recall@k increases consistently with k for all methods, while the relative ranking remains stable: Dense < BM25 < Hybrid. At k = 5, Recall@5 reaches 0.5119, 0.5812, and 0.6012 for Dense, BM25, and Hybrid, respectively. Similar trends are observed across other metrics, including nDCG and MRR.

Dense Retrieval consistently underperforms BM25 and Hybrid across all reported metrics, while Hybrid achieves the highest values on the FinQA dev set. In the current financial QA setting, this result shows that lexical matching remains important, while combining lexical and semantic signals provides a more suitable retrieval basis than relying on either signal alone. Because financial documents often contain both explicit terminology or numerical expressions and cross-sentence semantic dependencies, Hybrid Retrieval is adopted as the base architecture for the subsequent optimization experiments on the basis of the primary dev set comparison.

### *3.2. Efficiency–Effectiveness Comparison of Optimization Methods on FinQA*

Following the unified protocol, this section compares Grid Search, Random Search, and Bayesian Optimization (BO) under the same search space, objective function, and budget constraints. The goal is to examine whether lower-cost methods can approach high-quality configurations and whether their development-set advantages transfer to the test set.

#### 3.2.1. Effectiveness on the Dev Set

Under the unified search space and budget setting, the three optimization methods show limited but observable differences in dev set effectiveness. Table 4 reports the per-seed results, and Table 5 summarizes the method-level descriptive statistics. Grid Search achieves the highest dev objective, 0.5167, after exhaustively evaluating all 60 valid configurations, with the best setting $cs = 1000$, $ov = 50$, and $\alpha = 0.5$. By contrast, Random Search and Bayesian Optimization (BO), each restricted to 20 trials per run, still reach competitive dev set results. Across five seeds, the mean best dev objective is $0.5052 \pm 0.0064$ for Random Search and $0.5080 \pm 0.0080$ for BO. The corresponding average gaps to the Grid optimum are 0.0115 and 0.0087, respectively, while the hit rate is 1/5 for Random Search and 2/5 for BO.

**Table 4.** Per-seed comparison of optimization methods on the FinQA dev set.

| Method | Seed | Budget | Best Trial | Best Dev Obj. | Best Config (cs, ov, $\alpha$) | Runtime to Best (s) | Test Obj. | Dev → Test Drop |
|---|---|---|---|---|---|---|---|---|
| Grid | — | 60 | 52 | 0.5167 | (1000, 50, 0.5) | 8409.7 | 0.4641 | −0.0526 |
| Random | 42 | 20 | 8 | 0.5014 | (800, 0, 0.7) | 316.6 | 0.453 | −0.0484 |
| Random | 123 | 20 | 0 | 0.5027 | (1000, 50, 0.7) | 27 | 0.4583 | −0.0444 |
| Random | 7 | 20 | 5 | 0.5027 | (1000, 50, 0.7) | 238.2 | 0.4583 | −0.0444 |
| Random | 2024 | 20 | 10 | 0.5027 | (1000, 50, 0.7) | 422.1 | 0.4583 | −0.0444 |
| Random | 3407 | 20 | 7 | 0.5167 | (1000, 50, 0.5) | 744.9 | 0.4641 | −0.0526 |
| BO | 42 | 20 | 7 | 0.5014 | (800, 0, 0.7) | 288.9 | 0.453 | −0.0484 |
| BO | 123 | 20 | 1 | 0.5036 | (1000, 100, 0.5) | 92.5 | 0.4477 | −0.0559 |
| BO | 7 | 20 | 10 | 0.5167 | (1000, 50, 0.5) | 992.8 | 0.4641 | −0.0526 |
| BO | 2024 | 20 | 2 | 0.5014 | (800, 0, 0.7) | 116.4 | 0.453 | −0.0484 |
| BO | 3407 | 20 | 0 | 0.5167 | (1000, 50, 0.5) | 24.2 | 0.4641 | −0.0526 |

**Table 5.** Method-level summary statistics for dev set effectiveness under the low-budget setting.

| Method | Best Dev Obj. (Mean ± Std) [Min–Max] | Best Config (Mode) | Runtime to Best (s) (Mean ± Std) [Min–Max] | Test Obj. (Mean ± Std) [Min–Max] | Dev → Test Drop (Mean ± Std) | Gap to Grid Best (Mean ± Std) | Hit Rate |
|---|---|---|---|---|---|---|---|
| Grid | 0.5167 [single run] | (1000, 50, 0.5) | 8409.7 [single run] | 0.4641 [single run] | −0.0526 [single run] | 0.0000 | Exhaustive reference |
| Random | 0.5052 ± 0.0064 [0.5014–0.5167] | (1000, 50, 0.7), 3/5 seeds | 349.8 ± 264.1 [27.0–744.9] | 0.4584 ± 0.0039 [0.4530–0.4641] | −0.0468 ± 0.0037 | 0.0115 ± 0.0064 | 1/5 |
| BO | 0.5080 ± 0.0080 [0.5014–0.5167] | tied: (800, 0, 0.7) and (1000, 50, 0.5), each 2/5 seeds | 303.0 ± 397.8 [24.2–992.8] | 0.4564 ± 0.0074 [0.4477–0.4641] | −0.0516 ± 0.0032 | 0.0087 ± 0.0080 | 2/5 |

Note: For BO, the modal best configuration is tied between (800, 0, 0.7). and (1000, 50, 0.5), each appearing in 2 of 5 runs.

Figure 3 further shows that the best objective values of Random Search and BO remain concentrated within a relatively narrow range close to the Grid reference. Under the current discrete search space, competitive development-set results can still be obtained under limited budgets. At the same time, the overlap between the two low-budget methods is substantial, suggesting that the average advantage of BO over Random Search on the dev set is present but small.

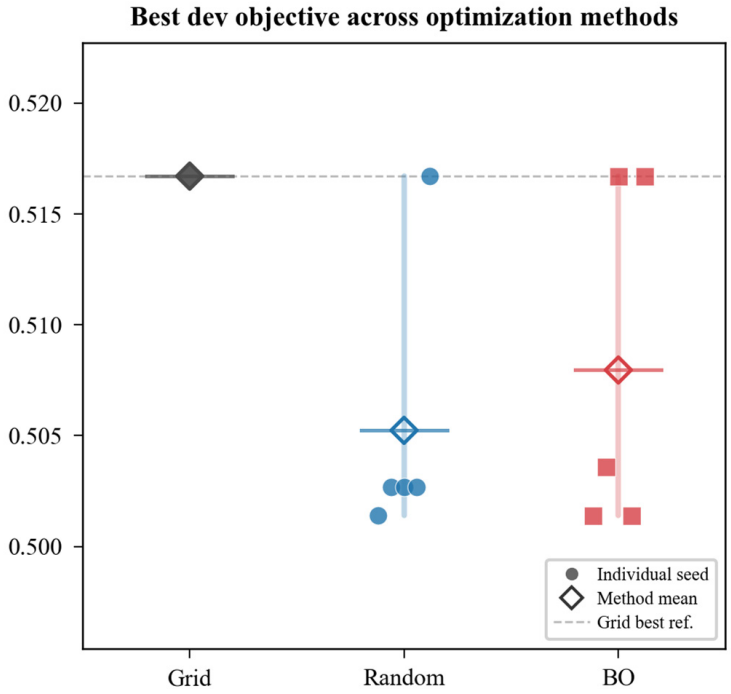



**Figure 3.** Development-set effectiveness of different optimization methods under the same search space. Note: gray, blue, and red denote Grid Search, Random Search, and Bayesian Optimization, respectively.

#### 3.2.2. Efficiency and Convergence Under Budget Constraints

The best dev result reflects only the search outcome and does not capture the cost required to obtain it. To address this issue, Figure 4 compares the three methods from two perspectives: search trajectories and runtime to best.

Grid Search achieves the highest dev objective, but it also incurs the largest search cost. It evaluates all 60 valid configurations, requires 8409.7 s to reach its best result, and does not identify the optimal configuration until Trial 52. By contrast, Random Search and BO are each limited to 20 trials per run and reach near-optimal dev results at substantially lower cost. Across five seeds, the mean runtime to best is 349.8 $\pm$ 264.1 s for Random Search and 303.0 $\pm$ 397.8 s for BO, indicating that both low-budget methods reduce search cost markedly relative to exhaustive search, although the runtime to best varies considerably across runs.

The search trajectories in Figure 4 further illustrate the difference in search behavior. Random Search shows a more dispersed pattern across trials, whereas BO enters high-performing regions earlier in some runs. However, this early-stage advantage is not consistent across all seeds, as reflected by the substantial variability in runtime to best.

Taken together, Grid Search provides the most complete coverage, while Random Search and BO achieve competitive dev set results at much lower search cost under the current budget setting.

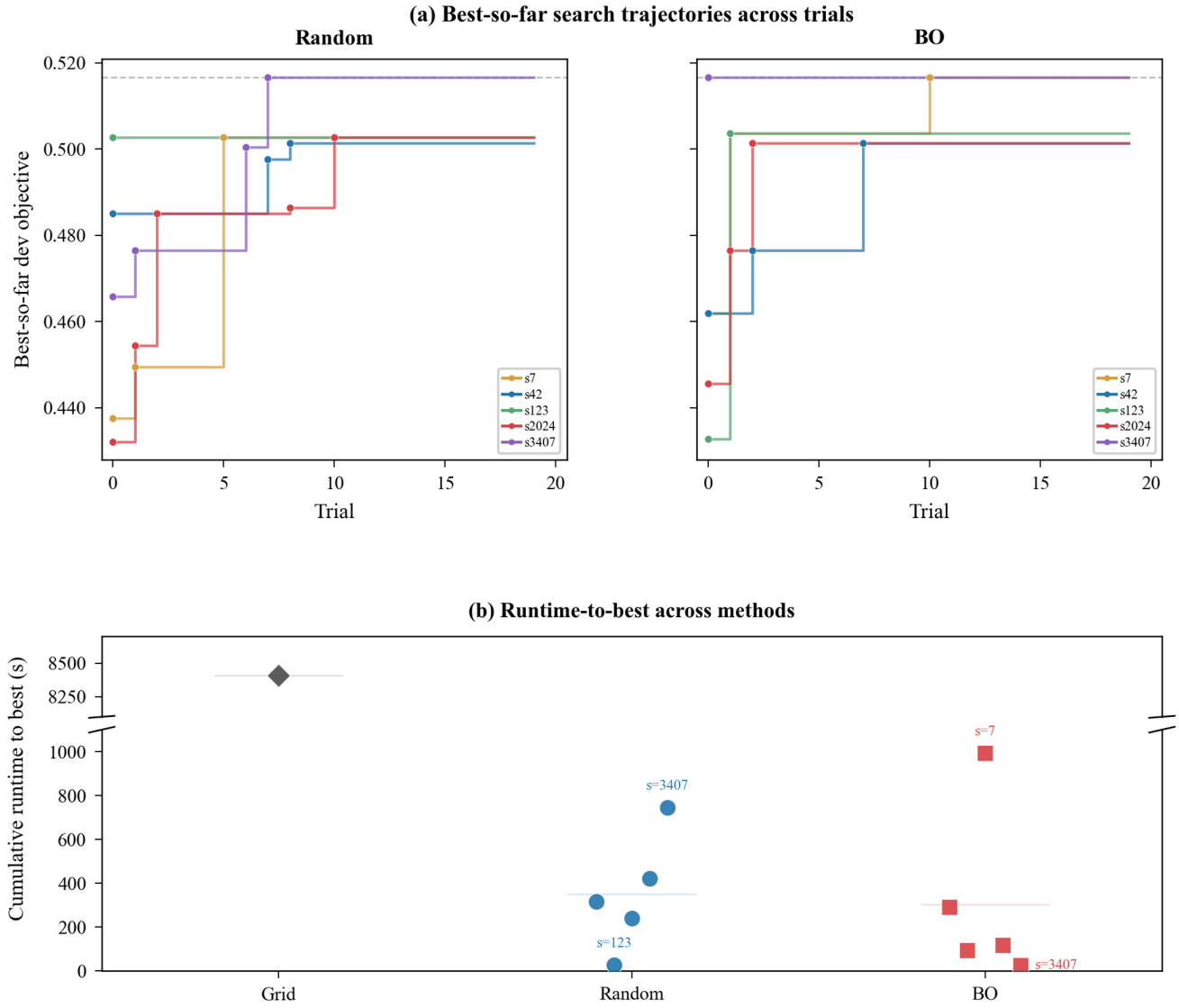



**Figure 4.** Search efficiency and convergence behavior under limited optimization budgets.

#### 3.2.3. Generalization to the Held-Out Test Set

Whether the small dev set differences among methods are practically meaningful depends on their transfer to an independent test set. Table 4 and Figure 5 therefore compare the dev-selected configurations with their corresponding held-out test performance. Across all methods, test results are lower than their dev counterparts, indicating a consistent generalization loss from dev to test. The Grid optimum still achieves the highest test objective, 0.4641.

For the two low-budget methods, the mean test objective is $0.4584 \pm 0.0039$ for Random Search and $0.4564 \pm 0.0074$ for BO. The corresponding mean dev-to-test drops are $-0.0468 \pm 0.0037$ and $-0.0516 \pm 0.0032$, respectively. Although BO reaches a slightly higher average dev objective, this difference becomes smaller on the test set. Random Search remains comparable and yields a slightly higher mean test objective. Figure 5 shows that the dev-to-test shift reduces the separation among methods under held-out evaluation.

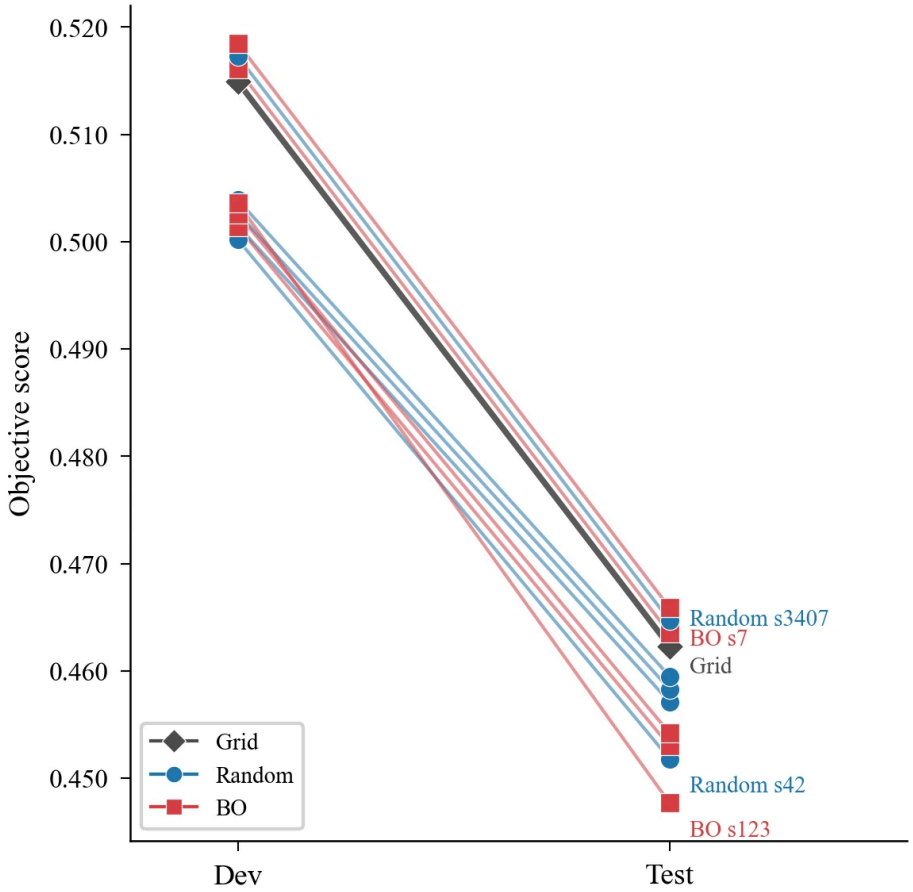



**Figure 5.** Transfer from development-selected configurations to held-out test performance.

### *3.3. Hyperparameter Sensitivity and Stable High-Performance Regions*

#### 3.3.1. Performance Trends Across Individual Hyperparameters

Figure 6 shows the marginal trends of the three hyperparameters across all evaluated trials. Within the current search space, chunk size exhibits the clearest trend: the mean objective generally increases with larger chunk sizes and reaches its highest level at $cs = 1000$. For chunk overlap, $ov = 50$ is associated with the highest mean objective, whereas $ov = 100$ performs less favorably. For the fusion weight, the mean objective increases from $\alpha = 0.1$ to $\alpha = 0.7$ and then declines at $\alpha = 0.9$. Overall, better results are concentrated in configurations with larger chunk sizes, moderate overlap, and medium-to-high fusion weights.

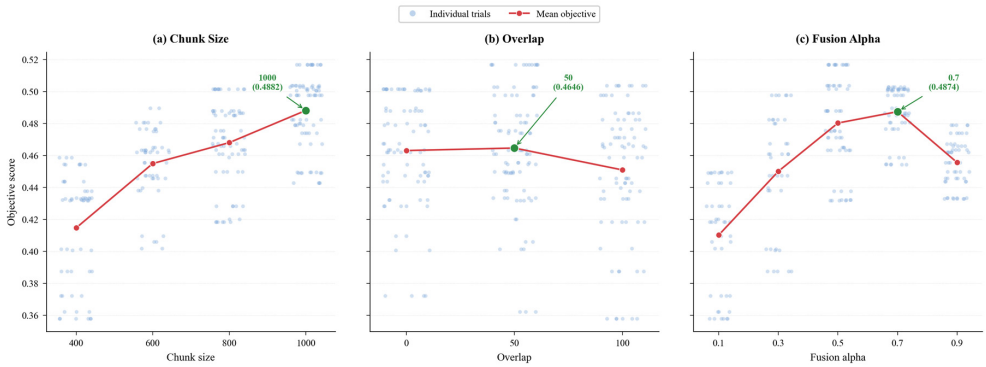



**Figure 6.** Performance trends across individual hyperparameters: (**a**) chunk size vs. objective; (**b**) overlap vs. objective; (**c**) fusion alpha vs. objective. The green arrow/annotation marks the highlighted parameter setting in each subplot.

The results show that performance is unevenly distributed across the search space rather than scattered uniformly. More detailed evidence is provided in the subsequent analysis of configuration concentration and stable high-performance regions.

#### 3.3.2. Configuration Concentration and Parameter Interactions

The best configurations identified across runs are concentrated in a limited set of repeated parameter combinations rather than dispersed across the search space. As reported in Table 4, only four unique best configurations are observed across the Grid reference and the ten low-budget runs. Among them, (1000, 50, 0.5), (1000, 50, 0.7), and (800, 0, 0.7) appear repeatedly, whereas (1000, 100, 0.5) is selected only once. Random Search most often selects $(1000, 50, 0.7)$, while BO repeatedly returns (800, 0, 0.7) and (1000, 50, 0.5). Repeated selections across runs show that high-performing configurations cluster in a relatively narrow part of the search space rather than appearing as isolated points.

Figure 7 further illustrates this concentration from the perspective of parameter interaction. Using the Grid Search results, the heatmap summarizes the mean objective value of each ($cs$, $ov$) pair after averaging over fusion weights. The higher-valued cells are concentrated in the lower part of the grid, indicating that larger chunk sizes are consistently associated with better average performance across overlap settings. Within this region, $ov = 50$ yields the highest mean objective at $cs = 1000$, while $ov = 0$ and $ov = 100$ remain competitive but slightly lower. By contrast, the upper part of the grid, especially at $cs = 400$, shows uniformly weaker performance.

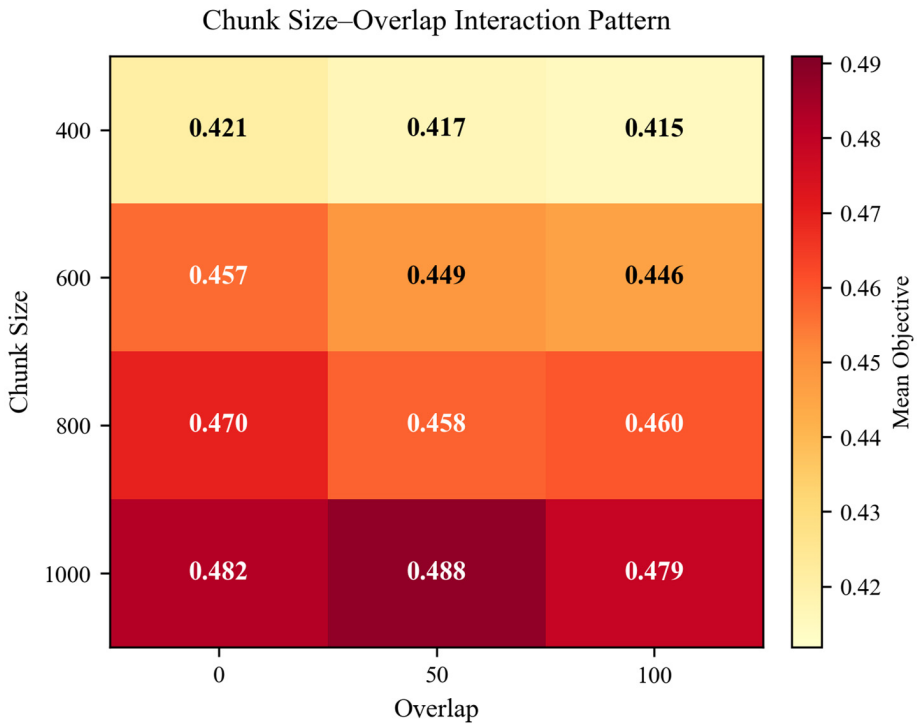



**Figure 7.** Mean objective across chunk size and overlap.

#### 3.3.3. Stable High-Performance Regions on Dev and Test

Figure 8 compares representative high-performing configurations on the development and test sets, and Table 6 reports their corresponding objective values and dev-to-test drops. On the dev set, the strongest configurations are concentrated around $cs = 1000$ with $\alpha = 0.5$ or 0.7, together with a nearby configuration at $cs = 800$ and $\alpha = 0.7$. The same region remains competitive on the test set. In particular, (1000, 50, 0.5) achieves the highest test objective, while (1000, 50, 0.7) and (800, 0, 0.7) also maintain similar

performance. By contrast, (1000, 100, 0.5) shows a larger dev-to-test drop than the other representative configurations.

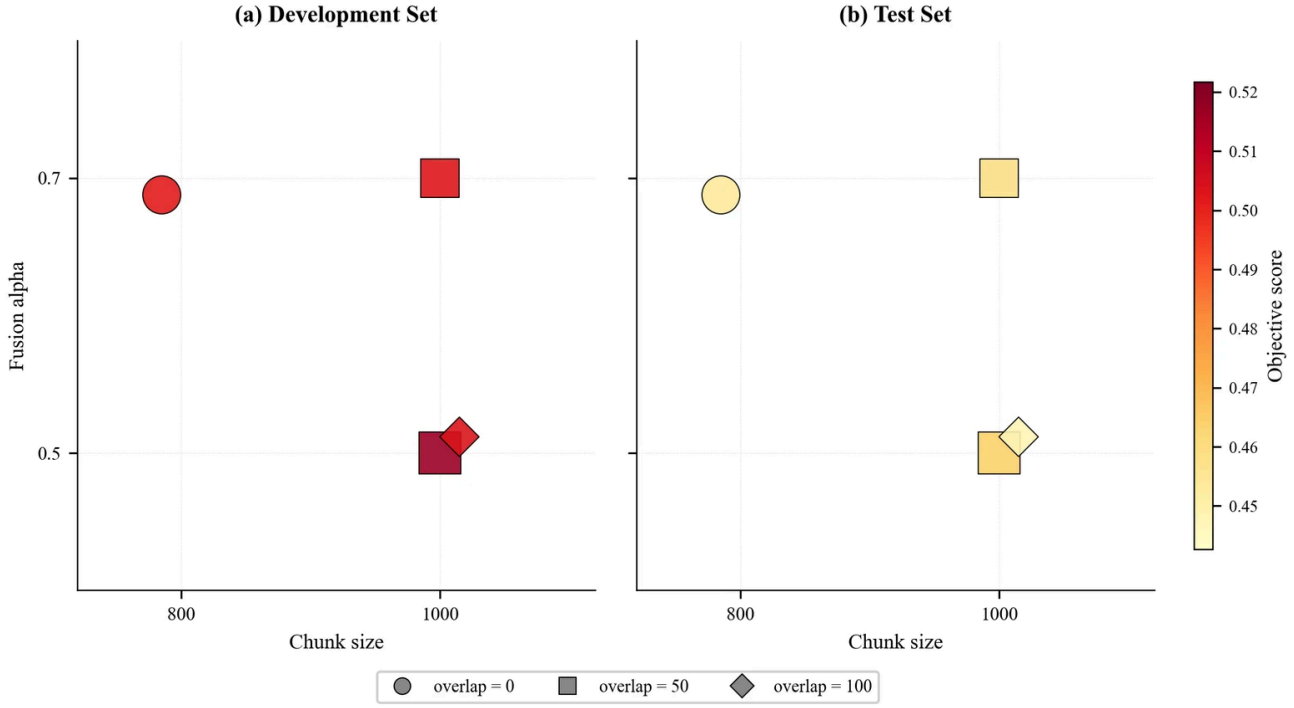



**Figure 8.** Stable high-performance regions on the development and test sets.

**Table 6.** Development-to-test comparison of representative high-performing configurations.

| Configuration | Dev Objective | Test Objective | Drop |
|---|---|---|---|
| cs = 1000, ov = 50, $\alpha$ = 0.5 | 0.5167 | 0.4641 | 0.0526 |
| cs = 1000, ov = 100, $\alpha$ = 0.5 | 0.5036 | 0.4477 | 0.0559 |
| cs = 1000, ov = 50, $\alpha$ = 0.7 | 0.5027 | 0.4583 | 0.0444 |
| cs = 800, ov = 0, $\alpha$ = 0.7 | 0.5014 | 0.453 | 0.0484 |

Overall, the dev–test comparison suggests that the relative differences among these representative configurations remain small, while configurations centered on larger chunk sizes and moderate overlap continue to perform competitively under held-out evaluation.

### *3.4. Statistical Significance and Multi-Seed Robustness*

#### 3.4.1. Architecture-Level Statistical Comparison Between BM25 and Hybrid

To examine whether the dev set advantage of Hybrid remains reliable on held-out evaluation, we compare BM25 and baseline Hybrid Retrieval on the FinQA test set under the same baseline setting ($chunk_size = 400, overlap = 50, \alpha = 0.5, top_k = 5$). Query-level nDCG@5 serves as the primary metric, and the comparison is conducted with a paired permutation test and a 95% bootstrap confidence interval for the mean paired difference (Hybrid − BM25). As shown in Table 7, BM25 and Hybrid achieve mean nDCG@5 values of 0.4115 and 0.4093, respectively, yielding a mean paired difference of −0.0023, with a 95% bootstrap confidence interval of $[-0.0181, 0.0128]$ and a permutation $p$-value of 0.7763.

The comparison shows that the difference is not statistically significant on the held-out test set. For Recall@5, Hybrid is only slightly higher than BM25 (0.5641 vs. 0.5634). Overall, the dev set advantage of Hybrid should not be interpreted as reliable evidence of superiority over BM25 in the baseline test-set setting. In this sense, Hybrid is used in the subsequent

experiments as the architecture selected on the basis of the primary dev set comparison, rather than as an architecture shown to be uniformly superior across evaluation settings.

**Table 7.** Architecture-level statistical comparison between BM25 and baseline Hybrid on the FinQA test set.

| Comparison | Mean BM25 nDCG@5 | Mean Hybrid nDCG@5 | Delta (Hybrid − BM25) | 95% Bootstrap CI | Permutation *p*-Value | Mean BM25 Recall@5 | Mean Hybrid Recall@5 |
|---|---|---|---|---|---|---|---|
| Baseline Hybrid vs. BM25 | 0.4115 | 0.4093 | −0.0023 | [−0.0181, 0.0128] | 0.7763 | 0.5634 | 0.5641 |

#### 3.4.2. Configuration-Level Statistical Comparison and Multi-Seed Robustness

This section compares the representative high-performing configurations on the test set and examines whether their differences remain stable across repeated runs. Figure 9a shows the per-query nDCG@5 differences for the three pairwise comparisons. In all three cases, the differences fluctuate around zero, indicating that no configuration consistently outperforms the others across queries. Table 8 and Figure 9b summarize the mean paired differences, 95% bootstrap confidence intervals, and Holm-adjusted *p*-values.

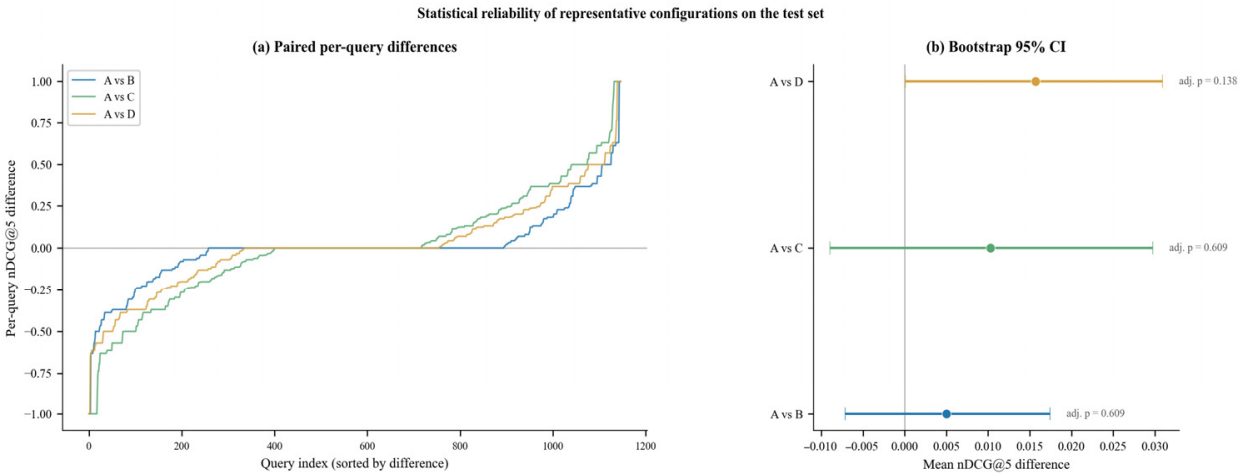



**Figure 9.** Statistical reliability analysis of representative configurations on the test set: (**a**) Paired per-query differences; (**b**) Bootstrap distribution.

**Table 8.** Bootstrap pairwise comparison results for representative configurations.

| Comparison | $Mean_1$ | $Mean_2$ | Delta | 95% CI | Raw *p* | Holm *p* |
|---|---|---|---|---|---|---|
| A vs. B | 0.4376 | 0.4326 | 0.005 | [−0.0072, 0.0174] | 0.4296 | 0.6087 |
| A vs. C | 0.4376 | 0.4273 | 0.0103 | [−0.0089, 0.0297] | 0.3044 | 0.6087 |
| A vs. D | 0.4376 | 0.4219 | 0.0157 | [0.0001, 0.0309] | 0.0459 | 0.1377 |

Note: A–D denote the four representative high-performing configurations compared on the held-out test set: A = (cs = 1000, ov = 50, k = 5, α = 0.5); B = (cs = 1000, ov = 50, k = 5, α = 0.7); C = (cs = 800, ov = 0, k = 5, α = 0.7); D = (cs = 1000, ov = 100, k = 5, α = 0.5).

As shown in Table 8, all three mean differences are positive. For A vs. B and A vs. C, the confidence intervals include zero. For A vs. D, the bootstrap confidence interval is slightly above zero, but the Holm-adjusted *p*-value remains above 0.05. Therefore, none of the pairwise differences remains statistically significant after multiple-comparison correction. These results indicate that the representative high-performing configurations achieve similar test-set performance.

The multi-seed results show the same pattern. Random Search and Bayesian Optimization repeatedly identify neighboring high-performing configurations across different seeds, and their test-set results remain close. What remains more consistent is the reproducibility of high-performing regions, rather than the superiority of any single configuration or optimization method.

### *3.5. External Validation on FinanceBench*

To examine whether the main findings extend beyond FinQA, we conduct external validation on FinanceBench. Under the same preprocessing principles and evaluation criteria, this dataset is used only for architecture-level comparison and representative-configuration transfer, rather than for full search-based optimization.

#### 3.5.1. Architecture-Level Validation

Table 9 reports the architecture-level results on FinanceBench under the baseline setting ($chunk_size = 400$, $overlap = 50$, $top_k = 5$, $\alpha = 0.5$). The ranking is Dense > Hybrid > BM25 across all reported metrics. Dense achieves the highest nDCG@5 (0.0951) and MRR (0.1774), followed by Hybrid (0.0740, 0.1308) and BM25 (0.0423, 0.0564). This ranking differs from the FinQA results, indicating that architecture-level performance is not fully transferable across datasets.

**Table 9.** Architecture-level validation results on FinanceBench under the baseline setting.

| Architecture | Recall@5 | MRR | nDCG@5 | Precision@5 |
|---|---|---|---|---|
| Dense | 0.0865 | 0.1774 | 0.0951 | 0.0720 |
| Hybrid | 0.0684 | 0.1308 | 0.0740 | 0.0547 |
| BM25 | 0.0384 | 0.0564 | 0.0423 | 0.0280 |

#### 3.5.2. Transferability of Representative Configurations

Figure 10 compares the FinanceBench baseline configuration with four representative high-performing configurations transferred from FinQA. All four transferred configurations outperform the baseline on FinanceBench. In terms of nDCG@5, the baseline reaches 0.0740, whereas the transferred configurations achieve values ranging from 0.1212 to 0.1684. The highest nDCG@5 among the transferred settings is 0.1684. MRR shows a similar pattern, with the best transferred configuration also achieving the highest MRR.

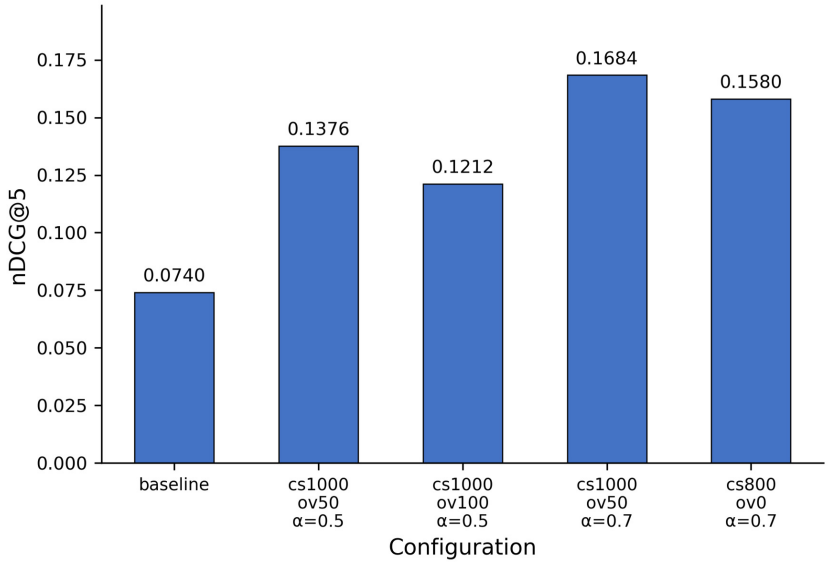



**Figure 10.** Representative configuration transfer on FinanceBench.

Taken together, the FinanceBench validation indicates that the representative configurations identified on FinQA remain competitive on the second dataset under the current setting, although the architecture-level ranking differs from that observed on FinQA.

## 4. Discussion

### *4.1. Efficiency–Effectiveness Trade-Off*

The differences among the three optimization methods mainly lie in search behavior and cost rather than in a decisive gap in final retrieval performance. Grid Search provides the most complete coverage of the predefined space, but at the highest cost. By contrast, Random Search and Bayesian Optimization (BO) approach the Grid reference under substantially lower budgets.

Their practical differences are therefore most visible in how they search the space. Grid Search offers exhaustive coverage but incurs the largest trial cost. Random Search provides a simple low-cost baseline with broad exploration, whereas BO uses historical observations to guide subsequent trials and can enter promising regions earlier in some runs. Under the current discrete and budget-constrained setting, the main practical value of more complex optimization lies in search efficiency rather than in a decisive difference in retrieval quality.

### *4.2. Stability of Optimization Advantages*

The small dev set advantage of BO does not translate into a stable held-out advantage. Random Search achieves comparable test performance, and the inferential results show that the differences among representative high-performing configurations do not remain statistically significant after multiple-comparison correction. The multi-seed results point to the same pattern, as Random Search and BO repeatedly identify neighboring configurations with similar downstream performance.

A plausible explanation lies in the structure of the current optimization setting. Under a low-budget, discrete search space, BO depends more heavily on early observations to guide subsequent trials. This can improve search efficiency in some runs, but it also makes the search path more sensitive to initial samples and local response patterns. BO is therefore better understood as offering a limited search-path advantage rather than robust evidence of superior final performance.

### *4.3. Stable High-Performance Hyperparameter Regions*

Compared with the instability of method-level differences, the repeated emergence of high-performing hyperparameter regions is a more robust finding. High-performing configurations are concentrated in a limited region characterized by larger chunk sizes, moderate overlap, and medium-to-high fusion weights, rather than scattered across the entire search space.

This pattern remains visible on both the development and test sets. Within the current search space, chunk size shows the clearest trend, while overlap and fusion weight mainly affect performance within the stronger region. A plausible explanation is that financial documents are often long and structurally dense, so larger chunks are more likely to preserve complete semantic units, whereas moderate overlap can help maintain contextual continuity without introducing excessive redundancy.

The external validation results support this interpretation. Although the architecture-level ranking changes on FinanceBench, the representative high-performing configurations transferred from FinQA remain competitive on the second dataset. On this basis, the more stable conclusion of this study concerns recurrent high-performance regions rather than one universally superior architecture or optimizer.

### *4.4. Limitations and Future Work*

Several limitations should be noted. First, the retrieval labels are automatically derived and should be regarded as approximate relevance annotations rather than noise-free ground truth. Second, although a second dataset is included for external validation, the cross-dataset evidence remains limited and does not support unrestricted generalization. Third, the current analysis does not examine whether the observed patterns remain stable across different embedding models. Finally, this study focuses on retrieval rather than end-to-end answer generation or deployment-level evaluation, and therefore does not test whether retrieval improvements directly translate into better final answer quality.

Future work may extend the analysis to broader financial benchmarks, compare alternative embedding settings, and examine whether the observed retrieval patterns remain consistent in downstream QA tasks.

## 5. Conclusions

This study examines black-box hyperparameter optimization for financial RAG retrieval under limited budget constraints. Using FinQA as the primary benchmark, it compares Grid Search, Random Search, and Bayesian Optimization within a unified search space, evaluation protocol, and multi-seed setting, and further uses FinanceBench for external validation.

However, the small development-set advantage of Bayesian Optimization does not remain stable on the held-out test set or across repeated runs. In other words, the relative superiority of more complex optimization methods is not robust under the current setting. A more stable finding is that high-performing configurations are concentrated in a limited region of the search space rather than scattered across isolated points.

Across the evaluated settings, larger chunk sizes, moderate overlap, and medium-to-high fusion weights are more often associated with competitive retrieval performance. The external validation results further suggest that architecture-level rankings may vary across datasets, whereas representative high-performing configurations remain competitive on the second dataset.

Taken together, the findings suggest that, in budget-constrained financial RAG retrieval tuning, the more practical goal is not simply to adopt increasingly complex optimizers, but to identify stable high-performing parameter regions under a transparent and reproducible evaluation protocol.

**Author Contributions:** Conceptualization, Y.J. and Q.D.; methodology, Q.D.; validation, Y.J. and Q.D.; formal analysis, X.W.; investigation, Y.J.; resources, Q.D. and Y.J.; data curation, Y.J.; writing—original draft preparation, Y.J. and Q.D.; writing—review and editing, Y.J. and Q.D.; visualization, Y.J.; supervision, X.W.; project administration, X.W.; funding acquisition, X.W. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by the College Students' Innovation and Entrepreneurship Training Program, Grant Number: 202610004005.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** The data presented in this study are derived from publicly available resources. FinQA is available through the $T^2$-RAGBench benchmark at https://huggingface.co/datasets/G4KMU/t2-ragbench (accessed on 14 April 2026), and the FinanceBench open-source subset is available from https://github.com/patronus-ai/financebench/tree/main# (accessed on 14 April 2026). The retrieval-oriented processed data generated in this study, including derived query–chunk relevance labels and evidence-to-chunk mappings, are available from the corresponding author upon reasonable request.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| BM25 | Best Matching 25 |
| BO | Bayesian Optimization |
| MRR | Mean Reciprocal Rank |
| nDCG | Normalized Discounted Cumulative Gain |
| QA | Question Answering |
| FinQA | Financial Question Answering |
| TPE | Tree-structured Parzen Estimator |
| RAG | Retrieval-Augmented Generation |

## References

1. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
2. Guu, K.; Lee, K.; Tung, Z.; Pasupat, P.; Chang, M.-W. Retrieval Augmented Language Model Pre-Training. In Proceedings of the 37th International Conference on Machine Learning, Virtual Event, 13–18 July 2020; Volume 119, pp. 3929–3938.
3. Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.-H.; Routledge, B.; et al. FinQA: A Dataset of Numerical Reasoning over Financial Data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Online and Punta Cana, Dominican Republic*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 3697–3711.
4. Islam, P.; Kannappan, A.; Kiela, D.; Qian, R.; Scherrer, N.; Vidgen, B. FinanceBench: A New Benchmark for Financial Question Answering. *arXiv* **2023**, arXiv:2311.11944. [CrossRef]
5. Strich, J.; Isgorur, E.K.; Trescher, M.; Biemann, C.; Semmann, M. $T^2$-RAGBench: Text-and-Table Benchmark for Evaluating Retrieval-Augmented Generation. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers), Rabat, Morocco*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2026; pp. 165–191.
6. Reddy, V.; Koncel-Kedziorski, R.; Lai, V.D.; Krumdick, M.; Lovering, C.; Tanner, C. DocFinQA: A Long-Context Financial Reasoning Dataset. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Bangkok, Thailand*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2024; pp. 445–458.
7. Kim, S.; Song, H.; Seo, H.; Kim, H. Optimizing Retrieval Strategies for Financial Question Answering Documents in Retrieval-Augmented Generation Systems. *arXiv* **2025**, arXiv:2503.15191. [CrossRef]
8. Lee, J.; Roh, M. Multi-Reranker: Maximizing Performance of Retrieval-Augmented Generation in the FinanceRAG Challenge. *arXiv* **2024**, arXiv:2411.16732.
9. Choe, J.; Kim, J.; Jung, W. Hierarchical Retrieval with Evidence Curation for Open-Domain Financial Question Answering on Standardized Documents. In *Findings of the Association for Computational Linguistics: ACL 2025, Vienna, Austria*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2025; pp. 16663–16681.
10. Izacard, G.; Grave, E. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume; Online*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2021; pp. 874–880.
11. Orbach, M.; Eytan, O.; Sznajder, B.; Gera, A.; Boni, O.; Kantor, Y.; Bloch, G.; Levy, O.; Abraham, H.; Barzilay, N.; et al. An Analysis of Hyper-Parameter Optimization Methods for Retrieval Augmented Generation. *arXiv* **2025**, arXiv:2505.03452. [CrossRef]
12. Bergstra, J.; Bardenet, R.; Bengio, Y.; Kégl, B. Algorithms for Hyper-Parameter Optimization. *Adv. Neural Inf. Process. Syst.* **2011**, *24*, 2546–2554.
13. Snoek, J.; Larochelle, H.; Adams, R.P. Practical Bayesian Optimization of Machine Learning Algorithms. *Adv. Neural Inf. Process. Syst.* **2012**, *25*, 2951–2959.
14. Bergstra, J.; Bengio, Y. Random Search for Hyper-Parameter Optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
15. Jimeno Yepes, A.; You, Y.; Milczek, J.; Laverde, S.; Li, R. Financial Report Chunking for Effective Retrieval Augmented Generation. *arXiv* **2024**, arXiv:2402.05131. [CrossRef]
16. Karpukhin, V.; Oguz, 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 (EMNLP)*; Association for Computational Linguistics: Stroudsburg, PA, USA, 2020; pp. 6769–6781.

17. Robertson, S.E.; Walker, S. Some Simple Effective Approximations to the 2-Poisson Model for Probabilistic Weighted Retrieval. In *SIGIR '94*; Springer: London, UK, 1994; pp. 232–241.
18. Hsu, H.-L.; Tzeng, J. DAT: Dynamic Alpha Tuning for Hybrid Retrieval in Retrieval-Augmented Generation. *arXiv* **2025**, arXiv:2503.23013. [CrossRef]
19. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-Generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA*; ACM: New York, NY, USA, 2019; pp. 2623–2631.
20. Järvelin, K.; Kekäläinen, J. Cumulated Gain-Based Evaluation of IR Techniques. *ACM Trans. Inf. Syst.* **2002**, *20*, 422–446. [CrossRef]
21. Voorhees, E.M.; Tice, D.M. The TREC-8 Question Answering Track. In *Proceedings of the Second International Conference on Language Resources and Evaluation (LREC'00), Athens, Greece*; European Language Resources Association (ELRA): Luxembourg, 2000.

**Disclaimer/Publisher's Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.