RAISE: RAG Design as an Architecture Search Problem

Anonymous ACL submission

Abstract

Retrieval-augmented generation (RAG) systems expose numerous design choices spanning query rewriting, chunking, retrieval depth, reranking, and context compression. In practice, these choices are often configured through heuristics, hindering systematic evaluation and reproducibility across settings. We argue that this challenge is best formulated as RAG architecture search. To support controlled and reproducible study of this problem, we introduce the RAG Intelligence Search Engine (RAISE), a comprehensive framework and benchmark for RAG hyperparameter optimization, which evaluates optimization methods for RAG pipelines under standardized search spaces and budgets. RAISE implements 13 search algorithms and evaluates them across seven public text and multimodal datasets using three random seeds. Our experiments show that optimization performance is highly task-dependent: methods that perform strongly on one dataset may not generalize consistently across others, cautioning against interpreting aggregate rankings as evidence of universally superior strategies. RAISE provides a common experimental substrate for fair, reproducible, and systematic research on RAG hyperparameter optimization.

1 Introduction

Retrieval-augmented generation (RAG) grounds LLM outputs in external knowledge (Lewis et al., 2020), but performance depends on tightly coupled choices such as chunker, query rewriting, retrieval depth, reranking, and context compression (Ammar et al., 2025). These choices are often set through heuristics or trial-and-error tuning, making RAG optimization costly, difficult to reproduce, and challenging to compare across systems.

Recent work has explored RAG configuration through AutoRAG (Kim et al., 2024) and hyperparameter impact analysis (Ammar et al., 2025), adaptive retrieval through Self-RAG (Asai et al., 2024) and Adaptive-RAG (Jeong et al., 2024), and automated tuning through AutoRAG-HP (Fu et al., 2024) and RAG-HPO analyses (Orbach et al., 2025). However, these efforts use different pipelines, datasets, search spaces, optimization methods, and evaluation protocols, leaving the field without a shared framework for comparing optimization algorithms under controlled conditions. Such a framework should expose search spaces explicitly, standardize evaluation protocols, and support matched computational budgets. It should also remain extensible to emerging optimization algorithms and benchmark tasks, similar to standardized environments used in hyperparameter optimization research such as HPOBench (Eggensperger et al., 2022) and YAHPO Gym (Pfisterer et al., 2022).

In this paper, we formulate RAG design as an architecture search problem. Instead of treating RAG tuning as an implementation detail, we formulate it as the selection of pipeline configurations that optimize end-to-end RAG performance under a fixed evaluation budget. This perspective connects RAG system design with random search (Bergstra and Bengio, 2012) and Bayesian optimization (Snoek et al., 2012), while retaining challenges specific to RAG: heterogeneous modules, coupled design choices, multimodal pipelines, and task-dependent objective landscapes.

To enable systematic study of this problem, we introduce the RAG Intelligence Search Engine (RAISE), a comprehensive framework and benchmark for RAG hyperparameter optimization (Figure 1). RAISE makes the problem reproducible and comparable across algorithms and settings, while supporting both LLM and multimodal LLM pipelines. The framework is designed to remain extensible beyond the benchmark tasks included in this work. New optimization algorithms can be integrated through a common controller API, while new benchmark tasks can be added by specifying

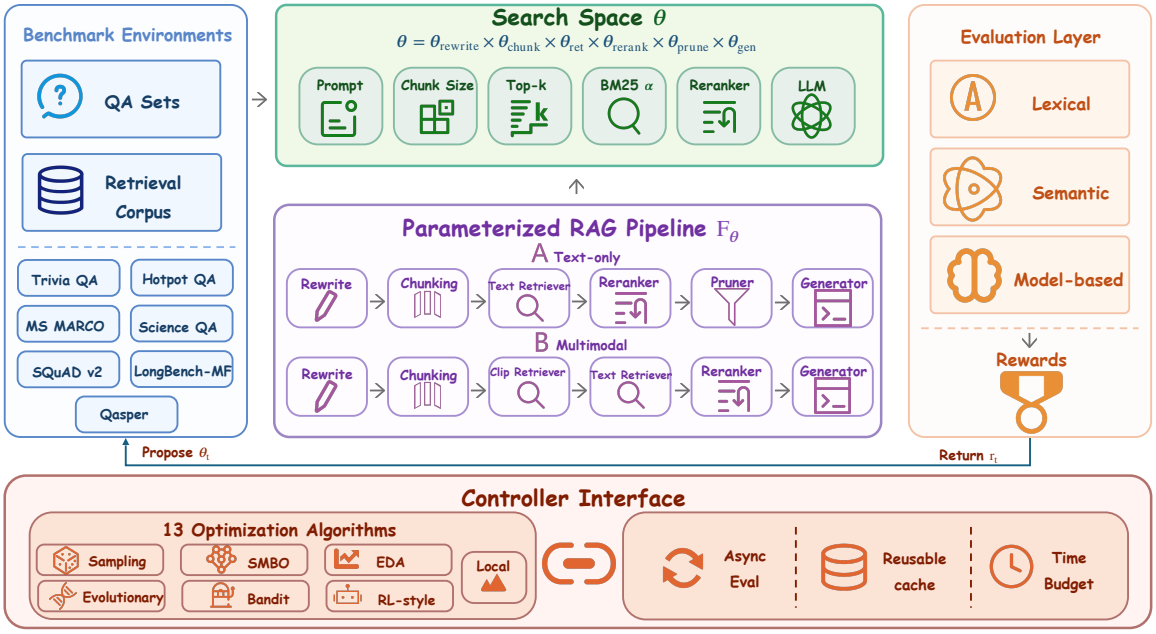

Figure 1: Overview of the RAG Intelligence Search Engine (RAISE). The framework couples a parameterized RAG pipeline, an evaluation layer that maps configurations to task-level rewards, and a controller interface for optimization algorithms.

datasets, corpora, and evaluation protocols without changes to the core pipeline implementation. We benchmark 13 optimization algorithms on seven environments: TriviaQA (Joshi et al., 2017), HotpotQA (Yang et al., 2018), MS MARCO (Bajaj et al., 2018), ScienceQA (Lu et al., 2022), SQuAD v2 (Rajpurkar et al., 2018), LongBench-Multifield (Bai et al., 2024), and LongBench-Qasper (Dasigi et al., 2021), with three seeds per setting.

We further conduct three-seed module ablations to identify which pipeline choices account for optimization gains under different task requirements. The results show that module effects vary with task structure: long-document retrieval benefits from query rewriting and pruning, while multi-hop reasoning relies more on retrieval depth. Likewise, long-context settings are especially sensitive to retrieval-depth control. These findings show that RAISE is not only a software framework but also a common experimental basis for studying RAG architecture search – an open problem we hope will attract broader research attention.

In summary, our contributions are as follows:

- We establish RAG architecture search as a benchmark setting, enabling existing and future optimization algorithms to be studied on end-to-end RAG pipelines through RAISE's unified interface. Our code will be released.
- We instantiate RAISE with a shared end-to-end RAG search space, public benchmark datasets, and a unified evaluation protocol. The benchmark spans query rewriting, chunking, retrieval, reranking, pruning, and generation, exposing heterogeneous task-specific optimization challenges under matched computational budgets.
- We conduct a controlled three-seed evaluation of 13 optimization algorithms across seven text and multimodal datasets, showing that optimizer performance is strongly environment-dependent. Our results motivate reporting RAG architecture search as optimizer–environment interactions rather than a universal leaderboard.

2 Related Work

RAG architectures and adaptive retrieval. Retrieval-augmented generation (RAG) was introduced as a means of combining parametric language models with external retrieval (Lewis et al., 2020). Subsequent work has shown that retrieval should often be adaptive rather than fixed: Self-RAG enables a model to decide when to retrieve and how to critique retrieved evidence during generation (Asai et al., 2024), while Adaptive-RAG

selects retrieval strategies based on question complexity (Jeong et al., 2024). Such studies confirm that end-to-end RAG quality hinges on retrieval and control decisions. Their focus, however, lies in designing or learning particular RAG architectures, rather than in general optimization over configurable RAG pipelines.

RAG hyperparameter optimization and benchmark environments. Several recent studies have begun to investigate hyperparameter optimization for RAG. AutoRAG-HP formulates RAG hyperparameter tuning as an online multi-armed bandit problem and employs a hierarchical controller for online adaptation (Fu et al., 2024). Orbach et al. (2025) compare several hyperparameter optimization methods for RAG across multiple datasets and a large search space. More broadly, our formulation builds on black-box and hyperparameter optimization, including random search (Bergstra and Bengio, 2012), Bayesian optimization (Snoek et al., 2012), budget-aware bandit methods such as Hyperband (Li et al., 2018), and hybrid approaches such as BOHB (Falkner et al., 2018). Benchmark suites such as HPOBench (Eggensperger et al., 2022) and YAHPO Gym (Pfisterer et al., 2022) further demonstrate the value of standardized environments for reproducible optimizer comparison. RAISE extends this benchmark perspective to end-to-end RAG pipelines, where coupled modules, expensive evaluations, and task-dependent objectives make controlled comparison especially important.

Benchmarking and evaluating RAG pipelines. A complementary line of work investigates how RAG systems should be evaluated. ARES decomposes RAG quality into context relevance, answer faithfulness, and answer relevance (Saad-Falcon et al., 2024), while RAGAs provides reference-free metrics for similar dimensions (Es et al., 2024). Benchmarks such as RGB (Chen et al., 2024) and CRUD-RAG (Lyu et al., 2025) further expose failure modes and component sensitivities in retrieval-augmented systems. Collectively, these studies establish that RAG quality is multi-dimensional and cannot be adequately captured by a single QA score. Our goal is complementary: rather than benchmarking RAG models or evaluation metrics in isolation, we benchmark optimization methods over shared end-to-end RAG configuration spaces under fixed budgets and standardized protocols.

Component	Hyperparameter	Values
Rewriter	prompt template	$\{\text{P1}, \text{P2}, \text{P3}\}$
Chunker	chunk size / chunk overlap	$\{256, 512, 1024, 2048\}$ / $\{0, 64, 128, 192\}$
Text Retriever	model / top-k / BM25 weight α	{all-MiniLM-L6-v2, all-MiniLM-L12-v2} / $\{1, 3, 5, 10, 20, 50\}$ / $\{0.0, 0.25, 0.5, 0.75, 1.0\}$
Reranker	model / top-k	{MiniLM-L6-v2, TinyBERT-L2-v2} / $\{1, 3, 5, 10, 20, 50\}$
LLM	LLM stack	Qwen3 series for rewriting, pruning, and generation
Multimodal LLM	Vision-LLM stack	Qwen3-VL series + CLIP top-k $\in \{1, 3, 5, 10, 20, 50\}$

Table 1: Instantiated experimental search space, with shared dimensions listed once and LLM and multimodal LLM components listed.

3 Methodology

We define RAG architecture search as optimizing a parameterized end-to-end RAG pipeline under a fixed evaluation protocol. RAISE realizes this view through three components: a pipeline abstraction that specifies the admissible configurations, an evaluation layer that maps configurations to task-level scores, and a controller interface through which optimization algorithms propose configurations and receive feedback from the environment. This separation lets us vary the search strategy while holding the pipeline family, benchmark data, and scoring rule fixed. We adopt the term *optimization algorithm* for the search method itself and *controller* for the same method once instantiated within RAISE.

3.1 Search Space and Pipeline Abstraction

RAISE models LLM and multimodal LLM RAG workflows as modular directed acyclic graphs. Formally, the search space Θ is the Cartesian product of module-specific configuration spaces:

$$\Theta = \Theta_{\text{rewrite}} \times \Theta_{\text{chunk}} \times \Theta_{\text{ret}} \times \Theta_{\text{rerank}} \times \Theta_{\text{prune}} \times \Theta_{\text{gen}} \quad (1)$$

This factorization disentangles query reformulation, chunk granularity, retrieval depth and scoring, reranking, pruning, and generation. LLM pipelines follow Rewriter $\rightarrow$ Chunker $\rightarrow$ Text Retriever $\rightarrow$ Reranker $\rightarrow$ Pruner $\rightarrow$ Generator, while multimodal LLM pipelines incorporate CLIP retrieval and omit pruning to preserve cross-modal

alignment. The text retriever combines sparse BM25 scoring (Robertson and Zaragoza, 2009) with dense sentence embeddings from Sentence-BERT (Reimers and Gurevych, 2019) and MiniLM (Wang et al., 2020), and the multimodal branch uses CLIP-aligned text–image embeddings (Radford et al., 2021). Detailed module definitions are provided in Appendix B.1.

3.2 Evaluation Objective

Given a dataset $\mathcal{D} = \{(q_i, Y_i^*)\}_{i=1}^N$ of queries and references, and a RAG pipeline parameterized by $\theta \in \Theta$, RAISE formulates the search problem via the following dataset-level objective:

$$\theta^* = \arg\max_{\theta \in \Theta} \frac{1}{N} \sum_{i=1}^{N} \mathcal{E}\left(\mathcal{F}_\theta(q_i), Y_i^*\right) \qquad (2)$$

where $\mathcal{F}_\theta(q_i) = Y_i$ denotes the generated response and $\mathcal{E}$ the evaluation function. Crucially, the optimization target is the full RAG pipeline rather than any module considered in isolation.

RAISE supports lexical, semantic, model-based, and efficiency-oriented signals, allowing $\mathcal{E}$ to be chosen for the task at hand. Table 1 gives the search space used in our experiments. In the main benchmark, we use an equal-weight objective over ROUGE-L (Lin, 2004), METEOR (Banerjee and Lavie, 2005), token-F1 (Rajpurkar et al., 2018), and BLEU (Papineni et al., 2002), as described in Section 4.2. Full metric definitions are provided in Appendix B.2.

3.3 Benchmark Environments

RAISE serves as both a configurable RAG pipeline and a benchmark environment for studying optimization behavior. Each environment is specified by a question–answer set and a retrieval corpus, keeping the interface simple while allowing tasks to differ in evidence structure, answer form, and modality. The suite covers six LLM QA tasks and one multimodal LLM QA task.

Benchmark construction. The benchmark is designed to expose bottlenecks in end-to-end RAG search rather than maximize dataset size. As shown in Table 2, the suite stresses long-document localization, multi-evidence composition, retrieval and reranking, abstention, long-context chunking and pruning, and visual grounding. The environments instantiate the seven datasets in Table 2.

Dataset	Type	QA / Corpus	Task	Primary pressure point
TriviaQA (Joshi et al., 2017)	Text	100 / 698	Open-domain	Long documents and alias-rich answers
HotpotQA (Yang et al., 2018)	Text	100 / 236	Multi-hop	Multi-evidence composition
MS MARCO (Bajaj et al., 2018)	Text	100 / 828	Retrieval	Retrieval and reranking quality
ScienceQA (Lu et al., 2022)	Text-Vision	100 / 100	Science	Visual-textual grounding
SQuAD v2 (Rajpurkar et al., 2018)	Text	100 / 100	Extractive	Hallucination control and abstention
LongBench-MF (Bai et al., 2024)	Text	100 / 100	Long-context	Cross-field retrieval and pruning
LongBench-Qasper (Dasigi et al., 2021)	Text	100 / 100	Scientific	Long-context reasoning and no-answer behavior

Table 2: RAISE evaluation suite instantiated as proxy environments for end-to-end search.

3.4 Supported Optimization Methods

RAISE includes 13 preset optimization algorithms grouped into seven descriptive families. The labels in Table 3 reflect the search state and selection rule used by each method, rather than a single canonical HPO taxonomy. We do not aim to tune a single optimizer for RAG. Instead, we use a common interface to expose different search biases. Algorithm-level details are given in Appendix B.4.

3.5 Controller Interface and Execution

Optimization algorithms use RAISE through a common controller interface. Let $\mathcal{A}$ denote a built-in or user-defined optimizer. Under this interface, $\mathcal{A}$ acts as a controller. At search iteration t, it proposes a complete pipeline configuration $\theta_t \in \Theta$, and the environment evaluates this configuration and returns a reward:

$$r_t = \text{RAISE_Env.evaluate}(\theta_t, \mathcal{D}) \qquad (3)$$

Here $r_t \in \mathbb{R}$, or a vector-valued reward in multi-objective settings, is computed by the evaluation layer on dataset $\mathcal{D}$.

This interface casts end-to-end RAG configuration as a black-box optimization problem under the same search space, datasets, budgets, and reward definition for all methods. The execution engine supports the practical requirements of large-scale

comparison, including asynchronous evaluation, reusable caches, bounded-time execution, and zero reward on failure, while keeping new controllers easy to add. Additional execution details are provided in Appendix B.3.

Algorithm	Family	Selection rule
Random Search (Bergstra and Bengio, 2012)	Random sampling	Uniformly samples full configurations
Greedy Search (Russell, 2010)	Local trajectory	Applies the best immediate local change
Coordinate Descent (Wright, 2015)	Local trajectory	Optimizes one dimension at a time
Simulated Annealing (Kirkpatrick et al., 1983)	Local trajectory	Accepts downhill moves under a temperature schedule
Iterative Local Search (Lourenco et al., 2001)	Local trajectory	Repeats local refinement from perturbed restarts
TPE (Bergstra et al., 2011)	SMBO/Bayes.	Samples from densities fitted to good and bad regions
Cross-Entropy Method (Rubinstein, 1999)	EDA-style	Updates a sampling distribution from elite configurations
Regularized Evolution (Real et al., 2019)	Evolutionary	Mutates selected parents in an aging population
Thompson Sampling (Thompson, 1933)	Bandit	Samples actions according to posterior uncertainty
UCB (Auer et al., 2002)	Bandit	Chooses actions by reward plus an exploration bonus
GRPO (Shao et al., 2024)	RL-style	Updates a policy with group-relative advantages
Dr. GRPO (Liu et al., 2025)	RL-style	Uses a more conservative GRPO-style update
Reinforce++ (Hu et al., 2025)	RL-style	Optimizes a stochastic policy with regularized policy gradients

Table 3: Thirteen preset optimization algorithms in RAISE, grouped by family and selection rule.

4 Experiments

4.1 Benchmark Protocol

We use RAISE as a controlled benchmark for RAG architecture search. The experimental design follows system-level RAG automation work, including AutoRAG (Kim et al., 2024), AutoRAG-HP (Fu et al., 2024), and RAG-HPO analysis (Orbach et al., 2025), that treats the full pipeline as the optimization target. In contrast, RAISE fixes the controller interface, evaluation budget, and benchmark suite so that optimizers can be compared under matched conditions, rather than being used only to produce a single tuned pipeline. This design allows us to examine whether a shared search space and heterogeneous environments reveal optimizer–environment interactions that are difficult to identify in isolated tuning studies. If RAG architecture search is governed by task structure, then the best-performing strategy should vary across retrieval-centric, multi-hop, abstention-sensitive, multimodal, and long-context environments. We therefore organize the empirical study around four research questions: whether optimizer rankings depend on the environment (RQ1), whether the proxy construction provides a sufficiently stable signal (RQ2), whether explicit search improves over random sampling under matched budgets (RQ3), and which pipeline modules and search dimensions have the largest effect on performance (RQ4).

We instantiate the seven environments in Table 2 as fixed proxy tasks from the corresponding public datasets, following HPOBench (Eggensperger et al., 2022) and YAHPO Gym (Pfisterer et al., 2022) benchmark practice. Each task contains 100 question–answer pairs and an associated retrieval corpus, yielding 700 QA instances and 2,162 corpus units in total. We evaluate the 13 optimization algorithms listed in Table 3 through a shared black-box interface, isolating optimizer behavior under matched conditions. Each algorithm receives 30 configuration evaluations under random seeds 11, 22, and 33. The scalar objective gives equal weight to ROUGE-L, METEOR, token-F1, and BLEU. LLM environments use Qwen3-4B-Instruct (Yang et al., 2025) for rewriting, pruning, and generation, while the multimodal LLM environment uses Qwen3-VL-4B-Instruct (Bai et al., 2025) with CLIP-based visual retrieval. Additional protocol details are provided in Appendix B.5.

4.2 Main Benchmark: Optimizer Environment Interactions (RQ1)

The main benchmark examines whether optimizer behavior in RAG hyperparameter optimization (RAG-HPO) is determined primarily by the optimization algorithm or by the structure of the RAG environment. To isolate this question, RAISE fixes the controller interface, evaluation budget, and scoring rule, while evaluating each optimizer on the corresponding text or multimodal search space

Family	Algorithm	Hotpot	MSM	SciQA	SQuAD	Trivia	Qasper	MF	Wins	Rank
Rand.	Random	0.353±0.043	0.232±0.014	**0.315±0.007**	0.223±0.009	0.401±0.006	0.093±0.005	0.336±0.023	1	5.6
Local	Greedy	**0.417±0.005**	0.199±0.013	0.311±0.006	0.223±0.004	**0.404±0.016**	0.092±0.005	0.339±0.044	2	5.7
Local	Coord.	0.409±0.020	0.231±0.037	0.313±0.004	0.224±0.004	0.397±0.023	0.094±0.002	0.337±0.036	0	4.6
Local	SA	0.368±0.038	0.205±0.017	0.266±0.062	0.200±0.017	0.402±0.009	0.092±0.007	**0.362±0.020**	1	6.9
Local	ILS	0.340±0.070	0.207±0.007	0.269±0.054	0.204±0.016	0.376±0.024	0.084±0.009	0.331±0.028	0	10.6
SMBO	TPE	0.359±0.072	0.221±0.003	0.269±0.029	0.226±0.004	0.387±0.027	0.101±0.007	0.362±0.002	0	5.1
EDA	CEM	0.323±0.014	**0.239±0.010**	0.230±0.075	0.225±0.002	0.390±0.021	0.090±0.008	0.348±0.012	1	7.4
Evol.	Reg-Evo	0.401±0.020	0.218±0.007	0.264±0.034	0.224±0.002	0.394±0.009	**0.102±0.005**	0.359±0.010	1	4.9
Bandit	TS	0.350±0.037	0.214±0.012	0.255±0.049	0.217±0.006	0.376±0.020	0.089±0.006	0.345±0.013	0	9.1
Bandit	UCB	0.307±0.065	0.205±0.023	0.248±0.044	0.202±0.012	0.389±0.017	0.089±0.008	0.334±0.032	0	11.4
RL-style	GRPO	0.333±0.057	0.214±0.027	0.288±0.031	**0.231±0.005**	0.402±0.017	0.086±0.008	0.349±0.011	1	6.3
RL-style	Dr. GRPO	0.380±0.049	0.219±0.030	0.276±0.049	0.202±0.008	0.395±0.003	0.095±0.006	0.348±0.013	0	6.0
RL-style	Reinforce++	0.372±0.025	0.236±0.010	0.252±0.082	0.212±0.014	0.394±0.009	0.095±0.005	0.328±0.013	0	7.4

Table 4: Main benchmark results as an optimizer–environment interaction table. All methods use the same 30-evaluation budget within each environment. Scores are computed with the RAISE main objective, the equal-weight average of ROUGE-L, METEOR, token-F1, and BLEU. Cells report the mean over three seeds with standard deviation. Bold marks the within-environment winner. *Wins* counts the number of dataset columns won by each method, and *Rank* is the average within-dataset rank, where lower is better.

for each environment. Under this protocol, differences across rows compare search behavior within the same environment, whereas differences across columns show how optimizer behavior changes across retrieval, reasoning, abstention, multimodal grounding, and long-context challenges.

Table 4 reports results on seven benchmark datasets: HotpotQA (Yang et al., 2018), MS MARCO (Bajaj et al., 2018), ScienceQA (Lu et al., 2022), SQuADv2 (Rajpurkar et al., 2018), TriviaQA (Joshi et al., 2017), LongBench-Qasper (Dasigi et al., 2021), and LongBench-Multifield (Bai et al., 2024). All runs use a budget of 30 trials under random seeds 11, 22, and 33. Each column corresponds to one environment and its primary challenge, and each row corresponds to an optimizer instantiated through the shared controller interface. Scores should be interpreted within each dataset column. The final two columns summarize the number of mean-score wins and the average within-dataset rank for each method. We therefore read the table as an optimizer–environment interaction map rather than as a single aggregate leaderboard. Detailed per-dataset seed results are provided in Appendix Tables 6–12.

Results and analysis. Table 4 shows that no optimizer dominates across environments. The best-performing method changes by task: Greedy Search leads on HotpotQA and TriviaQA, CEM on MS MARCO, Random Search on ScienceQA, GRPO on SQuADv2, Regularized Evolution on LongBench-Qasper, and Simulated Annealing on LongBench-Multifield. Under a fixed interface, budget, and scoring rule, these shifts indicate optimizer–environment interaction rather than a stable global ranking.

The aggregate columns support the same conclusion. Coordinate Descent achieves the best average rank without winning any individual dataset, while several other methods win only in specific settings. We therefore treat Table 4 as an interaction matrix that identifies where each search bias is useful.

Figure 2 shows that the optimizer–environment interaction in Table 4 also appears at the configuration level. Rewriting is frequently disabled, while retrieval, reranking, and pruning choices vary across methods and environments. Thus, comparable scores can arise from different pipeline choices, which motivates evaluating full configurations rather than isolated hyperparameters.

Figure 3 shows representative best-so-far trajectories over the same 30-trial budget.

4.3 Ablation Study

We complement the main benchmark with three ablation studies that examine the stability of proxy-task size and random seeds, the effect of explicit search relative to random sampling, and the contribution of individual pipeline modules to performance. Together, these analyses test whether

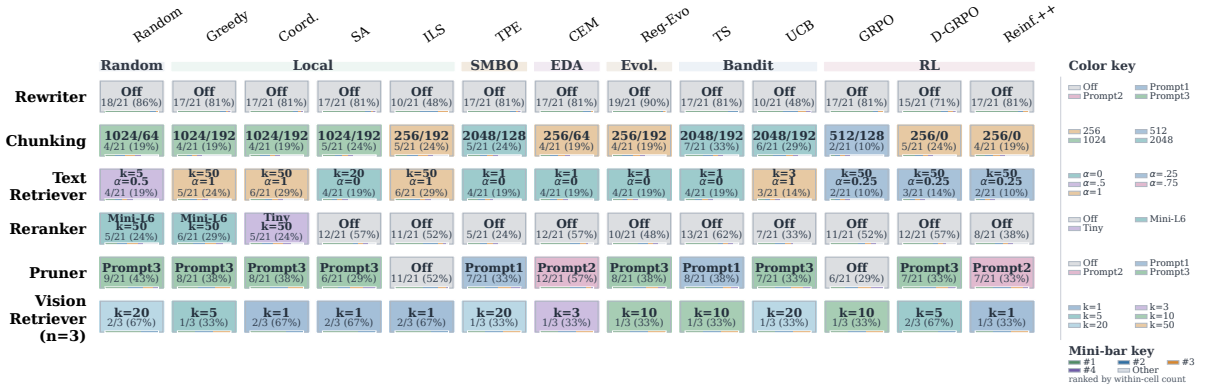

Figure 2: Module-option choices in the final best configurations, aggregated over seven environments and three seeds (N=21; ScienceQA-only vision retriever N=3). Colors indicate option frequency and disabled settings; mini-bars show top alternatives. For retrieval and reranking, k denotes top-k candidates and α denotes the BM25 weight. Full distributions are reported in Appendix A.

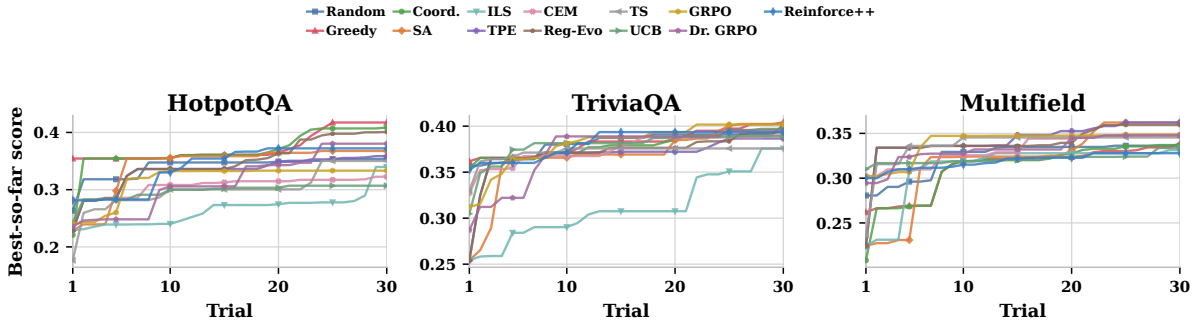

Figure 3: Example best-so-far search trajectories under a 30-trial budget, showing how controllers improve as RAISE evaluates additional configurations.

the main findings are robust to proxy construction and reveal which parts of the RAG pipeline most strongly shape optimization outcomes.

4.3.1 Robustness to QA Sample Size and Seeds (RQ2)

To examine how proxy-task size affects optimization stability, we vary the HotpotQA subset size from 20 to 5,000 examples for CEM and TPE, using five random seeds and 30 trials per run. Figure 4 shows substantial cross-seed variability for very small proxies, especially for TPE with 20 QA examples. At this scale, a small number of question instances can dominate the objective, making controller comparisons sensitive to seed selection. Stability improves at approximately 100–200 examples and then begins to saturate, while CEM remains more stable than TPE across the tested range. The flattening of the curves suggests that larger subsets mainly reduce noise once the proxy captures the main optimization signal. We therefore use the proxy as a controlled ranking screen rather than as a replacement for full-dataset evaluation. These results support the proxy protocol as a practical tradeoff between stability and cost for lightweight benchmarking, because it preserves the coarse optimizer signal without requiring full-corpus evaluation. This design keeps the benchmark lightweight while retaining enough task signal to compare optimizer behavior.

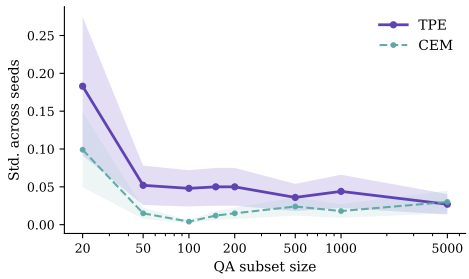

Figure 4: Proxy-size stability across QA subset sizes.

4.3.2 Explicit vs. Random Search (RQ3)

To test explicit optimization against unguided exploration, we compare search methods with random sampling on the HotpotQA proxy task under

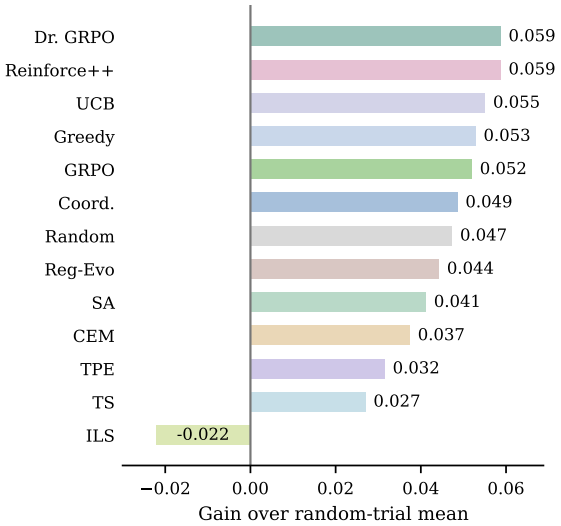

Figure 5: Algorithm comparison over three random trials on the HotpotQA proxy task. Bars report gains over the random-trial mean.

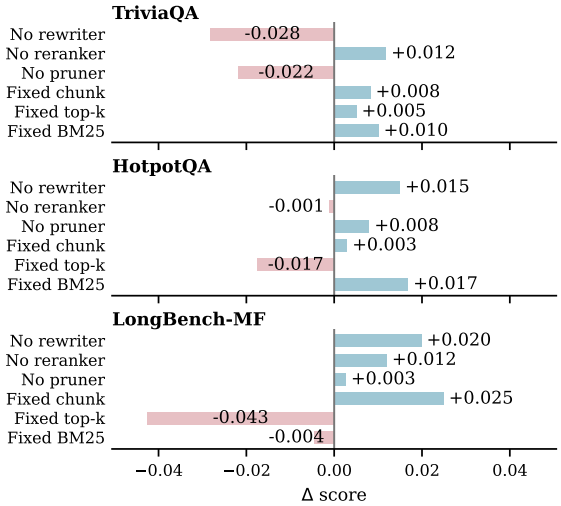

Figure 6: Module sensitivity on TriviaQA, HotpotQA, and LongBench-Multifield. Bars show mean best-score changes relative to the full pipeline.

the same 20-evaluation budget and weighted objective. Figure 5 shows that 11 of 12 non-random methods beat the baseline. The gains are modest but consistent, indicating that structured search can extract useful signal even when the budget is small. Adaptive methods tend to do better because they can spend later trials near more promising configurations. This advantage is budget-dependent rather than absolute. The result does not imply that random search is ineffective in general, but it shows that matched-budget comparisons should report random-trial baselines rather than relying only on optimizer rankings.

4.3.3 Impact of Pipeline Modules (RQ4)

We use module-level ablations on TriviaQA, HotpotQA, and LongBench-Multifield to examine how individual pipeline components affect performance under different task structures. Starting from the full text pipeline, we either remove the rewriter, reranker, or pruner, or fix one search dimension at a time. Each variant uses 13 optimization algorithms, three seeds, and 20 configuration evaluations.

Figure 6 shows that module effects vary substantially across datasets. TriviaQA is most sensitive to rewriting and pruning, consistent with its long-document retrieval setting and alias-rich answers. HotpotQA is more sensitive to retrieval depth because it requires multi-hop evidence composition. LongBench-Multifield shows the largest drop when retrieval top-k is fixed, whereas fixing chunk size is mildly beneficial. Across the three settings, no single module is uniformly dominant. Instead, the most influential search dimensions follow the evidence structure of the task, reinforcing the broader pattern that RAG pipeline choices are task-dependent. This finding cautions against treating any module choice as globally beneficial without specifying the environment and search budget used to evaluate it.

5 Conclusion

In this paper, we have presented RAISE, a framework and benchmark that formulates RAG-HPO as black-box optimization over complete RAG pipelines. Across seven environments and 13 algorithms, we have shown that optimizer rankings vary with task structure and that no method is uniformly best. Our ablations have shown that proxy construction, random baselines, and module choices affect how gains are interpreted.

RAISE has turned RAG tuning into a repeatable benchmark problem. By separating the controller, search space, and environment, it has provided a shared testbed for comparing optimizers while reducing pipeline and evaluation confounds. This separation has made mixed results informative, because they show where search behavior transfers and where it remains environment-specific under fixed protocols. Accordingly, future RAG-HPO studies should report the search space, proxy construction, random baseline, and module constraints alongside final scores for interpretable, reproducible cross-study comparisons.

Limitations

This study has several limitations. Although RAISE is extensible, our experiments instantiate a fixed search space over specific RAG modules, models, and discrete values; alternative model families, retrieval backends, or continuous spaces may alter algorithm behavior. The main benchmark relies on lightweight proxy environments and a fixed budget, which enables broad comparison but does not substitute for full-dataset or larger-budget studies. Rankings are based on an equal-weight lexical and token-level objective, with semantic and model-based metrics reported as auxiliary signals. Finally, the variance analysis covers only selected settings and warrants extension across a wider range of datasets, budgets, and seeds.

References

Adel Ammar, Anis Koubaa, Omer Nacar, and Wadii Boulila. 2025. Optimizing retrieval-augmented generation: analysis of hyperparameter impact on performance and efficiency. *arXiv preprint arXiv:2505.08445*.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. 2024. Self-rag: Learning to retrieve, generate, and critique through self-reflection. In *The Twelfth International Conference on Learning Representations*.

Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.*, 47(2–3):235–256.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, Wenbin Ge, Zhifang Guo, Qidong Huang, Jie Huang, Fei Huang, Binyuan Hui, Shutong Jiang, Zhaohai Li, Mingsheng Li, and 45 others. 2025. Qwen3-vl technical report. *Preprint*, arXiv:2511.21631.

Yushi Bai, Xin Lv, Jiajie Zhang, Hongchang Lyu, Jiankai Tang, Zhidian Huang, Zhengxiao Du, Xiao Liu, Aohan Zeng, Lei Hou, Yuxiao Dong, Jie Tang, and Juanzi Li. 2024. LongBench: A bilingual, multitask benchmark for long context understanding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3119–3137, Bangkok, Thailand. Association for Computational Linguistics.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. 2018. Ms marco: A human generated machine reading comprehension dataset. *Preprint*, arXiv:1611.09268.

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, Ann Arbor, Michigan. Association for Computational Linguistics.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems 24*, pages 2546–2554.

James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(10):281–305.

Jiawei Chen, Hongyu Lin, Xianpei Han, and Le Sun. 2024. Benchmarking large language models in retrieval-augmented generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17754–17762.

Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. 2021. A dataset of information-seeking questions and answers anchored in research papers. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4599–4610, Online. Association for Computational Linguistics.

Katharina Eggensperger, Philipp Müller, Neeratyoy Mallik, Matthias Feurer, René Sass, Aaron Klein, Noor Awad, Marius Lindauer, and Frank Hutter. 2022. Hpobench: A collection of reproducible multi-fidelity benchmark problems for hpo. *Preprint*, arXiv:2109.06716.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Stefan Falkner, Aaron Klein, and Frank Hutter. 2018. Bohb: Robust and efficient hyperparameter optimization at scale. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1437–1446. PMLR.

Jia Fu, Xiaoting Qin, Fangkai Yang, Lu Wang, Jue Zhang, Qingwei Lin, Yubo Chen, Dongmei Zhang, Saravan Rajmohan, and Qi Zhang. 2024. Autoraghp: Automatic online hyper-parameter tuning for retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 3875–3891.

Jian Hu, Jason Klein Liu, Haotian Xu, and Wei Shen. 2025. Reinforce++: Stabilizing critic-free policy optimization with global advantage normalization. *Preprint*, arXiv:2501.03262.

Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong C Park. 2024. Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7036–7050.

Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

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

Scott Kirkpatrick, Jr. C. Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by simulated annealing. *Science*, 220(4598):671–680.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474.

Lisha Li, Kevin Jamieson, Giulia DeSalvo, Afshin Rostamizadeh, and Ameet Talwalkar. 2018. Hyperband: A novel bandit-based approach to hyperparameter optimization. *Journal of Machine Learning Research*, 18(185):1–52.

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.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. 2025. Understanding r1-zero-like training: A critical perspective. *Preprint*, arXiv:2503.20783.

H. R. Lourenco, O. C. Martin, and T. Stutzle. 2001. Iterated local search. *Preprint*, arXiv:math/0102188.

Pan Lu, Swaroop Mishra, Tanglin Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and Ashwin Kalyan. 2022. Learn to explain: Multimodal reasoning via thought chains for science question answering. *Advances in neural information processing systems*, 35:2507–2521.

Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, and Enhong Chen. 2025. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *ACM Trans. Inf. Syst.*, 43(2).

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

Matan Orbach, Ohad Eytan, Benjamin Sznajder, Ariel Gera, Odellia Boni, Yoav Kantor, Gal Bloch, Omri Levy, Hadas Abraham, Nitzan Barzilay, Eyal Shnarch, Michael E. Factor, Shila Ofek-Koifman, Paula Ta-Shma, and Assaf Toledo. 2025. An analysis of hyper-parameter optimization methods for retrieval augmented generation. *Preprint*, arXiv:2505.03452.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Florian Pfisterer, Lennart Schneider, Julia Moosbauer, Martin Binder, and Bernd Bischl. 2022. Yahpo gym - an efficient multi-objective multi-fidelity benchmark for hyperparameter optimization. In *Proceedings of the First International Conference on Automated Machine Learning*, volume 188 of *Proceedings of Machine Learning Research*, pages 3/1–39. PMLR.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

Esteban Real, Alok Aggarwal, Yanping Huang, and Quoc V. Le. 2019. Regularized evolution for image classifier architecture search. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1):4780–4789.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson and Hugo Zaragoza. 2009. The probabilistic relevance framework: Bm25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Reuven Y. Rubinstein. 1999. The cross-entropy method for combinatorial and continuous optimization. *Methodology and Computing in Applied Probability*, 1(2):127–190.

Stuart J Russell. 2010. *Artificial intelligence a modern approach*. Pearson Education, Inc.

Jon Saad-Falcon, Omar Khattab, Christopher Potts, and Matei Zaharia. 2024. Ares: An automated evaluation framework for retrieval-augmented generation systems. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 338–354, Mexico City, Mexico. Association for Computational Linguistics.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *Preprint*, arXiv:2402.03300.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, pages 2951–2959.

William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3/4):285–294.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: deep self-attention distillation for task-agnostic compression of pre-trained transformers. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Ronald J. Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3–4):229–256.

Stephen J. Wright. 2015. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *Preprint*, arXiv:2505.09388.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. *Preprint*, arXiv:1904.09675.

A Module Preference Details

Figure 2 uses compact notation to keep the main-text matrix readable. In chunking cells, a/b denotes chunk size a and overlap b. In retrieval and reranking cells, $k = 5$ denotes top-5 candidates, and α denotes the BM25 weight in the hybrid text retriever. P1–P3 denote the discrete prompt templates used by the rewriter or pruner. The stacked bars below expand the mini-bars in the main figure.

B Detailed Framework Specification

B.1 Pipeline Module Definitions

This appendix provides the detailed mathematical specification of the parameterized pipeline modules summarized in Section 3.

B.1.1 Prompt Template Search Space

RAISE treats prompt selection as a discrete search dimension for the rewriter and pruner. The generator prompt is fixed across text experiments, and the LLM-as-a-judge prompt is used only for auxiliary evaluation. Table 5 lists the prompt templates exposed to the optimizer.

B.1.2 Query Rewriter (Both Modalities)

User queries in real-world scenarios are frequently ambiguous or underspecified. The Query Rewriter leverages LLMs to reformulate the user input prior to retrieval. Let $\mathcal{P} = \{p_1, p_2, \ldots, p_N\}$ denote a discrete search space comprising N candidate prompt strategies. Given an initial query q and a selected prompt template $p_i \in \mathcal{P}$, the rewriter operates as a conditional generative function producing the optimized query q':

$$q' \sim \mathcal{M}_{\text{rewrite}}(\cdot \mid q, p_i) \tag{4}$$

In RAISE, this categorical search space encompasses distinct reformulation paradigms, including standard retrieval reformulation, keyword or entity extraction, and standalone question generation.

B.1.3 Chunker Module (Both Modalities)

Text chunking critically affects retrieval granularity. Let $\mathcal{D}$ denote a document. The chunking function partitions $\mathcal{D}$ into a set of segments $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$, parameterized by chunk size s and overlap o:

$$\mathcal{C} = \text{Chunk}(\mathcal{D}; s, o) \tag{5}$$

In our implementation, the search space includes $s \in \{256, 512, 1024, 2048\}$ tokens and $o \in \{0, 64, 128, 192\}$ tokens.

Figure 7: Detailed module-option distributions behind Figure 2. Each panel corresponds to one RAG module, each bar corresponds to one optimizer, and stacked segments show the normalized frequency of final options among the best configurations. Standard text modules aggregate seven environments and three seeds ($n = 21$ per optimizer); the vision retriever is ScienceQA-only ($n = 3$ per optimizer).

B.1.4 Retrieval Stage: Text and Vision

The retriever serves as the first-stage filter that narrows the candidate corpus to a manageable subset.

Text Retriever (Both Modalities). RAISE employs hybrid retrieval mechanisms that blend sparse and dense signals. For a rewritten query q' and chunk $c \in \mathcal{C}$, the retrieval score is defined as

$$s(q', c) = \alpha \cdot \text{BM25}(q', c) + (1-\alpha) \cdot \cos(E_q(q'), E_d(c)) \quad (6)$$

where α controls the interpolation between lexical and semantic retrieval. The initial retrieved set is then

$$\mathcal{C}_{\text{ret}} = \underset{c \in \mathcal{C}}{\text{Top-}k_{\text{ret}}}\, s(q', c) \quad (7)$$

with search parameters $\alpha \in \{0.0, 0.25, 0.5, 0.75, 1.0\}$ and $k_{\text{ret}} \in \{1, 3, 5, 10, 20, 50\}$.

Vision Retriever (Multimodal Only). For multimodal tasks, RAISE incorporates a vision-language retrieval stage based on CLIP-style embeddings. Given a visual corpus $\mathcal{V}$, the retrieved image subset is

$$\mathcal{V}_{\text{ret}} = \underset{I \in \mathcal{V}}{\text{Top-}k_{\text{vision}}} \cos(E_{\text{text}}(q'), E_{\text{vision}}(I)) \quad (8)$$

which provides visual evidence in parallel with text retrieval.

B.1.5 Reranker (Both Modalities)

To refine the first-stage retrieval output, the reranker applies a cross-encoder that jointly scores the query and candidate document (Nogueira and Cho, 2020). Let $\mathcal{M}_{\text{cross}}$ denote the cross-encoder scoring function and $\oplus$ sequence concatenation. The reranked candidate set is

$$\mathcal{C}_{\text{rerank}} = \underset{c \in \mathcal{C}_{\text{ret}}}{\text{Top-}k_{\text{rerank}}}\, \mathcal{M}_{\text{cross}}(q' \oplus c) \quad (9)$$

where the search space encompasses both the reranker choice and $k_{\text{rerank}} \in \{1, 3, 5, 10, 20, 50\}$.

B.1.6 Pruner (Text Pipeline Only)

Even after reranking, accumulated evidence may exceed the generator's context window or introduce

Module	ID	Prompt template
Rewriter	P1	Rewrite the user query for retrieval. Output only the rewritten query; do not answer the question or add explanations.
Rewriter	P2	Rewrite the user query for retrieval with keywords and entities. Output only the rewritten query; do not answer the question or add explanations.
Rewriter	P3	Rewrite the user query as a standalone question for retrieval. Output only the rewritten query; do not answer the question or add explanations.
Pruner	P1	Keep only sentences that directly support the answer. Output only the pruned context text; do not answer the question or add explanations.
Pruner	P2	Select the minimal context needed to answer. Output only the pruned context text; do not answer the question or add explanations.
Pruner	P3	Remove irrelevant content and keep key evidence only. Output only the pruned context text; do not answer the question or add explanations.
Generator	fixed	Answer the question using only the provided context. If the answer is not in the context, state that it is unknown.
LLM judge	fixed	Judge whether the answer matches the reference list and return a JSON score with a short reason.

Table 5: Prompt templates used by RAISE. Rewriter and pruner prompts are selected through the search space; generator and judge prompts are fixed for controlled evaluation.

distracting content. Let $\mathcal{P}_{\text{prune}} = \{p_1, p_2, \ldots, p_M\}$ denote a discrete space of pruning prompt templates. The pruner applies a filtration function f_{prune} to produce a condensed context:

$$\mathcal{C}_{\text{final}} = f_{\text{prune}}(\mathcal{C}_{\text{rerank}}, q'; p_j) \tag{10}$$

where $p_j \in \mathcal{P}_{\text{prune}}$ specifies the pruning strategy. The multimodal pipeline bypasses this stage to preserve visual–textual alignment.

B.1.7 Generator

The generator constitutes the final synthesis stage. Given the textual context $\mathcal{C}_{\text{final}}$, the retrieved visual context $\mathcal{V}_{\text{ret}}$ for multimodal tasks, and the original query q, generation follows the autoregressive factorization

$$P(Y \mid q, \mathcal{C}_{\text{final}}, \mathcal{V}_{\text{ret}}) = \prod_{t=1}^{T} P_{\theta_{\text{gen}}}(y_t \mid y_{<t}, q, \mathcal{C}_{\text{final}}, \mathcal{V}_{\text{ret}}) \tag{11}$$

where $\mathcal{V}_{\text{ret}} = \emptyset$ in LLM pipelines.

B.2 Evaluation Metrics

This appendix provides the metric definitions summarized in Section 3.2. Let Y denote the predicted answer and Y^* the reference answer or reference context, depending on the task.

B.2.1 Lexical and N-gram Metrics

Exact Match (EM). For extractive question answering, exact match is defined as

$$\text{EM}(Y, Y^*) = \mathbb{I}(Y = Y^*) \tag{12}$$

Token-F1. Token-F1 treats Y and Y^* as bag-of-words and computes

$$F_1(Y, Y^*) = \frac{2 \cdot |Y \cap Y^*|}{|Y| + |Y^*|} \tag{13}$$

ROUGE-L. ROUGE-L (Lin, 2004) is based on the longest common subsequence (LCS), with

$$F_{\text{LCS}} = \frac{(1+\beta^2) R_{\text{LCS}} P_{\text{LCS}}}{R_{\text{LCS}} + \beta^2 P_{\text{LCS}}} \tag{14}$$

where $R_{\text{LCS}} = \frac{\text{LCS}(Y,Y^*)}{|Y^*|}$ and $P_{\text{LCS}} = \frac{\text{LCS}(Y,Y^*)}{|Y|}$.

BLEU. BLEU (Papineni et al., 2002) is computed as

$$\text{BLEU} = \text{BP} \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \tag{15}$$

METEOR. METEOR (Banerjee and Lavie, 2005) is defined as

$$\text{METEOR} = \left(\frac{10 P_m R_m}{R_m + 9 P_m}\right)(1 - \text{Pen}) \tag{16}$$

B.2.2 Semantic and Model-Based Metrics

BERTScore-F1. BERTScore (Zhang et al., 2020) computes semantic overlap using contextualized embeddings. Its recall term is

$$R_{\text{BERT}} = \frac{1}{|Y^*|} \sum_{y^* \in Y^*} \max_{y \in Y} \mathbf{y}^\top \mathbf{y}^* \tag{17}$$

LLM-as-a-Judge (LLM-aaJ). For complex reasoning tasks, RAISE supports an LLM-based judge model $\mathcal{M}$ whose output is parsed into a scalar reward:

$$\mathcal{E}_{\text{LLM}}(q, Y, R) = \Phi\left(\mathcal{M}(\text{prompt} \oplus q \oplus Y \oplus R)\right) \in \{0, 1\} \tag{18}$$

B.2.3 System and Search Efficiency

End-to-End Search Time. The total wall-clock time of the search process is

$$\mathcal{T}_{\text{total}} = \sum_{k=1}^{K} t_k + \mathcal{T}_{\text{overhead}} \tag{19}$$

Configuration Evaluation Count. The total number of completed evaluations over S search iterations is

$$K = \sum_{s=1}^{S} \mathbb{I}(\text{eval}(\theta_s)) \tag{20}$$

B.3 Execution Details and Robustness

Evaluating large numbers of configurations over full datasets incurs substantial computational overhead. To support large-scale search, the RAISE execution engine employs asynchronous concurrent execution to improve throughput, reusable caching to avoid redundant computation, and bounded-time evaluation to prevent individual failures from stalling the optimization loop.

Specifically, each proposed configuration is evaluated under explicit resource and time limits. Configurations that fail due to runtime errors, latency spikes, or resource exhaustion are assigned a zero reward, ensuring that the global search process remains well-defined. The environment additionally supports both corpus-level reward aggregation, suitable for standard black-box optimizers, and finer-grained feedback appropriate for reinforcement-learning methods.

B.4 Algorithm Details

This appendix provides additional details for the 13 algorithms summarized in Table 3.

B.4.1 Sampling and Local Search

Random Search. Random Search (Bergstra and Bengio, 2012) samples complete configurations independently from the discrete search space, providing the reference baseline for all structured optimizers.

Greedy Search. Greedy Search starts from a current configuration and applies the locally best modification available at each step. It quantifies how far the search space can be navigated by pure exploitation.

Coordinate Descent. Coordinate Descent (Wright, 2015) optimizes one hyperparameter dimension while holding the others fixed, and is most effective when the objective is partially separable across modules.

Simulated Annealing. Simulated Annealing (Kirkpatrick et al., 1983) performs local moves but accepts lower-scoring configurations with a temperature-controlled probability, allowing the search to escape poor local optima early and become more selective later.

Iterative Local Search. Iterative Local Search (Lourenco et al., 2001) alternates between local improvement and explicit perturbation. Relative to single-trajectory local search, it reduces dependence on initialization.

B.4.2 SMBO, Distribution-Guided, and Evolutionary Search

Tree-structured Parzen Estimator (TPE). TPE is an SMBO method (Bergstra et al., 2011) related to Bayesian optimization (Snoek et al., 2012). It partitions observed configurations into high- and low-reward sets and fits separate density models to the two regions; new candidates are then selected to favor configurations with high expected improvement under this surrogate.

Cross-Entropy Method. The Cross-Entropy Method (Rubinstein, 1999) is treated here as an estimation-of-distribution-style optimizer: it maintains a parameterized sampling distribution over configurations and updates it from the current elite set. In discrete spaces, this yields a direct model of which choices should accumulate more probability mass.

Regularized Evolution. Regularized Evolution (Real et al., 2019) maintains a finite population, samples parents from it, and generates offspring through mutation. Aging-based regularization removes old individuals and prevents premature population collapse.

B.4.3 Bandit Search

Thompson Sampling. Thompson Sampling (Thompson, 1933) chooses actions by sampling

from posterior reward estimates, yielding stochastic exploration proportional to uncertainty. In our setting, it serves as a bandit-style baseline for adaptive selection.

Upper Confidence Bound (UCB). UCB (Auer et al., 2002) selects the action with the largest optimistic score

$$\text{Score}_{\text{UCB}} = \bar{r}_i + c\sqrt{\frac{\ln N}{n_i}}, \tag{21}$$

where $\bar{r}_i$ is the empirical mean reward of action i, n_i its pull count, and N the total number of decisions. The rule renders the exploration term explicit and deterministic.

B.4.4 RL-based Methods

GRPO. GRPO (Shao et al., 2024) treats configuration generation as a policy over discrete choices and updates the policy with group-relative advantages. This reduces variance relative to raw score-based updates and enables direct policy learning from black-box rewards.

Dr. GRPO. Dr. GRPO (Liu et al., 2025) is a GRPO variant designed to reduce optimization bias and improve token efficiency. In our benchmark, we use it as a conservative GRPO-style baseline for noisy, small-budget settings.

Reinforce++. Reinforce++ (Hu et al., 2025) optimizes a stochastic policy with critic-free, globally normalized policy-gradient updates. It serves as a lighter RL baseline against which the more specialized GRPO variants can be compared, while remaining closely related to classical REINFORCE-style updates (Williams, 1992).

B.5 Benchmark Protocol Details

The main algorithm comparison is conducted on the seven benchmark tasks listed in Table 2. Each algorithm is evaluated under the corresponding instantiated text or multimodal search space, with seeds 11, 22, and 33 and a uniform budget of 30 configuration evaluations per seed. This protocol isolates search behavior from changes in pipeline structure, dataset size, or metric definition, while reducing dependence on a single random seed.

The scalar objective is an equal-weight aggregate of ROUGE-L, METEOR, token-F1, and BLEU. In LLM settings, rewriting, pruning, and generation are performed by the Qwen3 text stack. All algorithms interact with the same black-box `evaluate(config)` interface and observe the same proxy dataset for each task.

This shared protocol is intended to keep the main comparison focused on search behavior rather than on metric selection or implementation differences. It thus enables controlled comparison across algorithm families while holding the end-to-end RAG environment fixed.

These settings are intentionally modest: the proxy benchmark keeps evaluation fast enough for repeated optimization while still exposing differences between controllers, and 30 evaluations per seed provides a fixed budget for comparing search efficiency. Using three seeds helps separate algorithmic effects from sampling noise, while the shared `evaluate(config)` interface ensures that all methods face the same pipeline implementation and scoring rule. Together, these choices make the benchmark suitable for controlled comparison rather than for absolute system tuning.

B.6 Reproducibility Details

Each benchmark environment is stored as a pair of files. The QA file contains query and reference fields, while the corpus file contains the retrieval units used by the pipeline. For LongBench-Multifield, LongBench-Qasper, and ScienceQA, the proxy subsets are sampled with a fixed subset seed of 42. ScienceQA additionally keeps examples with a valid answer, a non-empty hint, and an available image, and exports the image path into the corpus entry. These fixed proxy files are used unchanged across all algorithms and seeds.

All text generation calls use the same local LLM configuration: temperature 0, maximum output length 256 tokens, and a 60-second request timeout. LLM runs use Qwen3-4B-Instruct for rewriting, pruning, and generation, while the multimodal LLM run uses Qwen3-VL-4B-Instruct for generation with CLIP-based visual retrieval. Experiments were run on a two-GPU workstation with PRO6000 GPUs.

A configuration evaluation means running one complete pipeline configuration over the full proxy environment and scoring the resulting answers with the selected objective. The evaluation cache is keyed by the configuration, dataset file hashes, modality, and evaluation mode, so repeated configurations return the same cached reward. Failed configurations receive zero reward and remain part of the budgeted trial sequence.

C Detailed Results

C.1 Detailed Main Results

Tables 6–12 report dataset-level algorithm results for the main benchmark. Each algorithm is evaluated with a budget of 30 trials under seeds 11, 22, and 33. The ranking score is the equal-weight aggregate of ROUGE-L, METEOR, token-F1, and BLEU. The tables report each seed separately and summarize the mean and standard deviation across seeds.

For compactness, we adopt the following abbreviations: Coord. = Coordinate Descent, SA = Simulated Annealing, ILS = Iterative Local Search, CEM = Cross-Entropy Method, Reg-Evo = Regularized Evolution, TS = Thompson Sampling, and Dr. GRPO = Dr. GRPO.

C.2 Random-Average Ablation Details

Table 13 reports the full per-algorithm results for the HotpotQA random-average ablation. All runs use a single seed and a budget of 20 evaluations. The random-trial mean baseline is 0.0660, and the Random Search best-of-budget score is 0.1134; the latter is included only as a reference point and is not the ablation baseline. The Δ column is always computed against the random-trial mean. BERTScore-F1 uses the corrected evaluation values.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
Greedy	0.4109	0.4205	0.4207	0.4174	0.0046
Coord.	0.3798	0.4205	0.4255	0.4086	0.0205
Reg-Evo	0.4064	0.3746	0.4217	0.4009	0.0196
Dr. GRPO	0.4071	0.4219	0.3121	0.3803	0.0487
Reinforce++	0.4035	0.3430	0.3701	0.3722	0.0247
SA	0.3254	0.3605	0.4171	0.3677	0.0378
TPE	0.3998	0.2568	0.4193	0.3586	0.0724
Random	0.3252	0.4145	0.3208	0.3535	0.0432
TS	0.3828	0.2992	0.3687	0.3502	0.0366
ILS	0.3777	0.3996	0.2417	0.3397	0.0698
GRPO	0.3968	0.3442	0.2587	0.3332	0.0569
CEM	0.3140	0.3114	0.3425	0.3226	0.0141
UCB	0.2759	0.2479	0.3969	0.3069	0.0647

Table 6: Three-seed detailed results for HotpotQA. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
CEM	0.2451	0.2248	0.2469	0.2389	0.0100
Reinforce++	0.2483	0.2350	0.2247	0.2360	0.0097
Random	0.2465	0.2366	0.2129	0.2320	0.0141
Coord.	0.2255	0.2791	0.1882	0.2309	0.0373
TPE	0.2204	0.2241	0.2172	0.2206	0.0028
Dr. GRPO	0.2483	0.2310	0.1768	0.2187	0.0305
Reg-Evo	0.2224	0.2242	0.2082	0.2183	0.0072
TS	0.2220	0.2241	0.1970	0.2144	0.0123
GRPO	0.1774	0.2219	0.2417	0.2137	0.0269
ILS	0.2100	0.2127	0.1974	0.2067	0.0067
SA	0.2101	0.1829	0.2226	0.2052	0.0166
UCB	0.2181	0.1729	0.2236	0.2049	0.0227
Greedy	0.1910	0.2179	0.1882	0.1990	0.0134

Table 7: Three-seed detailed results for MS MARCO. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
GRPO	0.2372	0.2288	0.2268	0.2309	0.0045
TPE	0.2297	0.2199	0.2280	0.2258	0.0043
CEM	0.2256	0.2225	0.2269	0.2250	0.0018
Reg-Evo	0.2209	0.2256	0.2265	0.2243	0.0025
Coord.	0.2231	0.2288	0.2192	0.2237	0.0040
Greedy	0.2217	0.2279	0.2191	0.2229	0.0037
Random	0.2160	0.2171	0.2356	0.2229	0.0090
TS	0.2079	0.2199	0.2226	0.2168	0.0064
Reinforce++	0.1939	0.2156	0.2273	0.2123	0.0139
ILS	0.2252	0.1987	0.1866	0.2035	0.0161
UCB	0.1916	0.1956	0.2196	0.2023	0.0124
Dr. GRPO	0.2126	0.1957	0.1966	0.2017	0.0078
SA	0.2006	0.1789	0.2195	0.1996	0.0166

Table 8: Three-seed detailed results for SQuADv2. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
Greedy	0.4195	0.3820	0.4104	0.4040	0.0160
SA	0.3949	0.4151	0.3969	0.4023	0.0091
GRPO	0.4206	0.4054	0.3792	0.4017	0.0171
Random	0.4072	0.4037	0.3934	0.4015	0.0059
Coord.	0.4030	0.3665	0.4214	0.3970	0.0228
Dr. GRPO	0.3917	0.3994	0.3951	0.3954	0.0032
Reg-Evo	0.3872	0.3879	0.4059	0.3937	0.0087
Reinforce++	0.4018	0.3812	0.3975	0.3935	0.0089
CEM	0.3648	0.4163	0.3893	0.3901	0.0210
UCB	0.4018	0.3648	0.4006	0.3891	0.0171
TPE	0.3525	0.3886	0.4185	0.3865	0.0270
TS	0.3525	0.3727	0.4020	0.3757	0.0203
ILS	0.4064	0.3492	0.3714	0.3756	0.0236

Table 9: Three-seed detailed results for TriviaQA. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
Reg-Evo	0.0990	0.1092	0.0971	0.1018	0.0053
TPE	0.0935	0.1098	0.0989	0.1007	0.0068
Dr. GRPO	0.0870	0.1009	0.0964	0.0948	0.0058
Reinforce++	0.0944	0.1010	0.0883	0.0946	0.0052
Coord.	0.0929	0.0930	0.0976	0.0945	0.0022
Random	0.0985	0.0917	0.0873	0.0925	0.0046
SA	0.0963	0.0974	0.0826	0.0921	0.0067
Greedy	0.0926	0.0849	0.0976	0.0917	0.0052
CEM	0.1019	0.0866	0.0828	0.0904	0.0082
TS	0.0829	0.0874	0.0980	0.0894	0.0063
UCB	0.0784	0.0901	0.0973	0.0886	0.0078
GRPO	0.0745	0.0918	0.0918	0.0861	0.0082
ILS	0.0956	0.0730	0.0824	0.0837	0.0092

Table 10: Three-seed detailed results for LongBench-Qasper. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
SA	0.3385	0.3883	0.3597	0.3622	0.0204
TPE	0.3600	0.3641	0.3621	0.3621	0.0017
Reg-Evo	0.3465	0.3613	0.3701	0.3593	0.0097
GRPO	0.3579	0.3552	0.3331	0.3487	0.0111
CEM	0.3316	0.3511	0.3616	0.3481	0.0124
Dr. GRPO	0.3570	0.3295	0.3565	0.3477	0.0129
TS	0.3314	0.3429	0.3621	0.3455	0.0127
Greedy	0.3729	0.2760	0.3677	0.3389	0.0445
Coord.	0.3576	0.2857	0.3677	0.3370	0.0365
Random	0.3522	0.3030	0.3535	0.3363	0.0235
UCB	0.3314	0.2958	0.3738	0.3337	0.0319
ILS	0.3634	0.2944	0.3358	0.3312	0.0284
Reinforce++	0.3092	0.3365	0.3380	0.3279	0.0133

Table 11: Three-seed detailed results for LongBench-Multifield. Scores are best-of-30 weighted objectives.

Alg.	Seed 11	Seed 22	Seed 33	Mean	Std.
Random	0.3244	0.3101	0.3101	0.3149	0.0067
Coord.	0.3180	0.3101	0.3101	0.3127	0.0037
Greedy	0.3182	0.3101	0.3046	0.3110	0.0056
GRPO	0.2451	0.3101	0.3101	0.2884	0.0307
Dr. GRPO	0.2063	0.3102	0.3101	0.2755	0.0490
ILS	0.1935	0.3044	0.3098	0.2693	0.0536
TPE	0.2515	0.3099	0.2461	0.2692	0.0289
SA	0.1784	0.3099	0.3101	0.2661	0.0620
Reg-Evo	0.2536	0.2280	0.3101	0.2639	0.0343
TS	0.3242	0.2217	0.2197	0.2552	0.0488
Reinforce++	0.1371	0.3102	0.3101	0.2525	0.0816
UCB	0.3102	0.2181	0.2152	0.2478	0.0441
CEM	0.3188	0.1346	0.2371	0.2302	0.0753

Table 12: Three-seed detailed results for ScienceQA. Scores are best-of-30 weighted objectives.

Alg.	Score	Δ	BERT	BLEU	F1	Judge	MET.	R-L
Dr. GRPO	0.1247	+0.0588	0.4388	0.0364	0.1475	0.5400	0.1712	0.1439
Reinforce++	0.1247	+0.0588	0.4388	0.0364	0.1475	0.5400	0.1712	0.1439
UCB	0.1211	+0.0551	0.4463	0.0371	0.1438	0.5300	0.1611	0.1425
Greedy	0.1189	+0.0529	0.4401	0.0368	0.1419	0.4800	0.1580	0.1387
GRPO	0.1180	+0.0520	0.4379	0.0330	0.1367	0.5500	0.1695	0.1329
Coord.	0.1146	+0.0487	0.4361	0.0292	0.1338	0.4800	0.1637	0.1317
Random	0.1134	+0.0474	0.4390	0.0305	0.1362	0.5400	0.1542	0.1325
Reg-Evo	0.1103	+0.0443	0.4360	0.0303	0.1313	0.5400	0.1503	0.1293
SA	0.1072	+0.0413	0.4369	0.0263	0.1265	0.5200	0.1521	0.1240
CEM	0.1034	+0.0375	0.4160	0.0319	0.1184	0.4700	0.1458	0.1176
TPE	0.0976	+0.0316	0.4175	0.0317	0.1096	0.4500	0.1406	0.1084
TS	0.0930	+0.0271	0.4083	0.0289	0.1034	0.4900	0.1384	0.1013
ILS	0.0437	-0.0222	0.3404	0.0046	0.0407	0.3600	0.0918	0.0378

Table 13: Full results for the random-average ablation on HotpotQA.