AutoRAG-HP: Automatic Online Hyper-Parameter Tuning for Retrieval-Augmented Generation

Jia Fu[1,2*], Xiaoting Qin[3], Fangkai Yang[3], Lu Wang[3], Jue Zhang[3†], Qingwei Lin[3], Yubo Chen[1,2], Dongmei Zhang[3], Saravan Rajmohan[3], Qi Zhang[3]

[1] Institute of Automation, Chinese Academy of Sciences, Beijing, China

[2] School of Artificial Intelligence, University of Chinese Academy of Sciences

[3] Microsoft

fujia2021@ia.ac.cn, yubo.chen@nlpr.ia.ac.cn

{xiaotingqin, fangkaiyang, wlu, juezhang, qlin}@microsoft.com

Abstract

Recent advancements in Large Language Models have transformed ML/AI development, necessitating a reevaluation of AutoML principles for the Retrieval-Augmented Generation (RAG) systems. To address the challenges of hyper-parameter optimization and online adaptation in RAG, we propose the AutoRAG-HP framework, which formulates the hyper-parameter tuning as an online multi-armed bandit (MAB) problem and introduces a novel two-level Hierarchical MAB (Hier-MAB) method for efficient exploration of large search spaces. We conduct extensive experiments on tuning hyper-parameters, such as top-k retrieved documents, prompt compression ratio, and embedding methods, using the ALCE-ASQA and Natural Questions datasets. Our evaluation from jointly optimization all three hyper-parameters demonstrate that MAB-based online learning methods can achieve Recall@5 ≈ 0.8 for scenarios with prominent gradients in search space, using only $\sim 20\%$ of the LLM API calls required by the Grid Search approach. Additionally, the proposed Hier-MAB approach outperforms other baselines in more challenging optimization scenarios. The code will be made available at https://aka.ms/autorag.

1 Introduction

Recent advancements in Large Language Models (LLMs) (Brown et al., 2020; Ouyang et al., 2022; OpenAI, 2023) represent a significant shift in the development of ML/AI solutions. Traditionally, scenario-specific models were trained for most ML/AI applications. However, in the LLM era, foundational models serve as the base, with supplementary modules added for practical applications. This transformation significantly affects the automation of ML/AI solution development, previously known as AutoML (Hutter et al., 2019; Bergstra et al., 2011a), necessitating a reevaluation of AutoML concepts in the context of LLMs.

Retrieval-Augmented Generation (RAG) has emerged as a prominent framework for building ML/AI solutions with LLMs (Lewis et al., 2020). While the standard RAG framework includes an information retrieval component to ground LLM's output in relevant data, numerous variants now integrate additional modules such as query rewriting (Ma et al., 2023), prompt compression (Jiang et al., 2023a; Pan et al., 2024), and query routing (Ding et al., 2024) to enhance performance.

The increased complexity of RAG systems presents two main challenges. First, the multitude of modules and hyper-parameters within the modules complicates the identification of optimal settings. Second, as we often receive online feedback from users (e.g., via thumb up/down feature), effectively utilizing those feedback to continuously tune the system is also crucial.

To address these challenges, we propose the development of an autonomous and self-optimizing system for RAG, termed **AutoRAG**, in line with the principles of AutoML. As the first step, this study focuses on hyper-parameter tuning in RAG (**AutoRAG-HP**). While there exist prior works discussing hyper-parameter tuning in RAG, they tend to either focus on tunable hyper-parameters in LLM API calls (Wang et al., 2023a) or assess the performance and impacts of RAG hyper-parameters through manual tuning (Lyu et al., 2024). In this work we focus on the optimization methods that can be applied in the online fashion. Specifically, we frame hyper-parameter selection as an online multi-armed bandit (MAB) problem (Lai and Robbins, 1985; Vermorel and Mohri, 2005; Li et al., 2010) and explore several variants in MAB. Moreover, to efficiently explore large search space when tuning multiple hyper-parameters simultaneously, we introduce a novel two-level Hierarchical MAB (Hier-MAB) method, wherein a high-level MAB

[*] Work is done during an internship at Microsoft.

[†] Corresponding author.

Findings of the Association for Computational Linguistics: EACL 2024, pages 3875–3891

guides the optimization of modules, while several low-level MABs search for optimal settings within each module. Our evaluation demonstrates that the MAB-based online learning methods are effective for scenarios with prominent gradients in search space, and the proposed Hier-MAB approach outperforms other baselines in more challenging optimization scenarios.

Our contributions can be summarized as follows:

- We introduce the AutoRAG-HP framework to address the pressing needs for optimal hyper-parameter tuning in RAG. To our best knowledge, we are the first to discuss the automatic online hyper-parameter tuning in RAG.
- We formulate the online hyper-parameter search in RAG as a multi-armed bandit problem and propose a novel two-level hierarchical multi-armed bandit method to efficiently explore large search space.
- The efficacy of our approach is validated across several scenarios using public datasets.

2 Related Work

2.1 AutoML and LLMs

In the process of developing ML/AI solutions, AutoML (Hutter et al., 2019; Bergstra et al., 2011a) has streamlined automation across three key areas: feature engineering, model construction, and hyper-parameter optimization. Over the past decade, AutoML has achieved remarkable success with heavily-utilized open-source frameworks like AutoSklearn (Feurer et al., 2015), TPOT (Olson et al., 2016), Auto-Keras (Jin et al., 2023), Auto-PyTorch (Zimmer et al., 2021), and FLAML (Wang et al., 2019). However, with the emergence of LLMs, a notable shift has occurred where LLMs are frequently chosen as base models, bypassing the traditional model design and training stages.

The research community has started to investigate the opportunities and challenges of applying AutoML to optimize pre-training, fine-tuning, and inference processes in the lifecycle of LLMs. A recent paper (Tornede et al., 2024) presents a timely survey discussing the potential symbiotic relationship between AutoML and LLMs, while also providing a future-oriented vision. Particularly, in leveraging AutoML for LLMs, existing efforts have primarily focused on hyper-parameter optimization during the pre-training and fine-tuning stages (Liu and Wang, 2021; Treviso et al., 2022; Tornede et al., 2024). LLaMA-NAS (Sarah et al., 2024) also explored efficient neural architecture search for LLMs. For the inference stage, EcoOptiGen (Wang et al., 2023a) represents a pioneering step towards applying AutoML to optimize LLM inference for text generation. This work targets tuning of hyper-parameters in OpenAI completion headers like temperature and max tokens. Another work (Pryzant et al., 2023) explored gradient decent and MAB in automatic optimization of prompts.

2.2 Hyper-parameter Optimization

Common hyper-parameter optimization techniques include methods such as Grid Search (Lecun et al., 1998), Random Search (Bergstra and Bengio, 2012a), Bayesian Optimization (Bergstra et al., 2011b) and manual tuning to identify optimal hyper-parameters. Grid search involves exhaustive searching within a predefined hyper-parameters grid, testing every possible combination to find the best fit. While straightforward, this approach can incur substantial computational costs, especially with expansive hyper-parameters spaces. Random search selects hyper-parameters through randomized sampling, yet its results may lack stability. Manual tuning, on the other hand, adjusts hyper-parameters based on domain knowledge or experience. While flexible, this method is time-consuming and challenging to standardize. These hyper-parameter optimization approaches often overlook the evaluation costs, particularly in evaluating solutions based on LLMs. BlendSearch (Wang et al., 2021) introduces an economic budget to enhance cost efficiency. However, these methods are not inherently learning-based and may not be well-suited for scenarios requiring adaptive optimization over time.

2.3 Hyper-parameter Tuning in RAG

Significant attention has been directed towards refining models within individual modules such as indexing, retrieval, and generation independently (Izacard et al., 2022; Jiang et al., 2023b; Ma et al., 2023; Wang et al., 2023b). However, in terms of hyper-parameters, these studies typically only report those leading to the best results, often chosen through manual tuning by experts during experimentation. Consequently, there has been scant exploration aimed at tuning hyper-parameters within each module, let alone collectively tuning various hyper-parameters across RAG modules.

CRUD-RAG (Lyu et al., 2024) has delved into the manual tuning of RAG hyper-parameters and assessed the performance and impacts of different components of the RAG system, such as the retriever and context length. While such studies offer valuable insights for optimizing RAG technology, their applicability across diverse scenarios or real-world applications is limited. Additionally, a project (Marker-Inc-Korea, 2024) mentions optimization via a greedy approach, initially generating all possible combinations of modules and hyper-parameters in each node.

3 Methodology

3.1 Problem Formulation

We formulate the hyper-parameter tuning problem in RAG as a multi-armed bandit (MAB) problem (Lai and Robbins, 1985), drawing an analogy to the scenario of a player selecting from a slot machine with multiple arms in a casino. The player's objective is to choose the arm that offers the highest expected gain. Each time the player pulls an arm and receives a gain or not, they update their estimation of the arm's potential gain. The MAB problem involves making sequential decisions, requiring the agent to balance the exploration of different arms to learn their reward probabilities and the exploitation of arms that are expected to yield higher rewards based on past observations. Given that MAB is an online learning method, it is well-suited for the online setting of AutoRAG-HP.

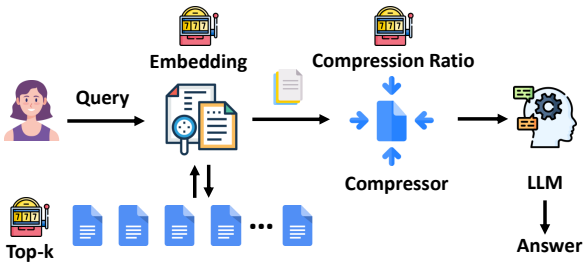

Figure 1: A RAG system with tunable hyper-parameters.

As an illustration, we present an example RAG system in Figure 1, which comprises a retrieval module, a prompt compression module, and a prompt construction module (not shown) that assembles the final prompt sent to LLMs for answer generation. In the retrieval module, we introduce two tunable hyper-parameters: the top-k ($\mathcal{K}$) document chunks retrieved from an external knowledge base and the embedding model ($\mathcal{E}$) used for ranking these retrieved chunks. With the top-k chunks retrieved, the prompt compression module then compresses tokens in each chunk to eliminate irrelevant information and save token cost (Jiang et al., 2023a; Pan et al., 2024). Since excessive compression may also remove relevant information, leading to decreased performance, it is crucial to find an optimal compression ratio ($\mathcal{C}$). Below, we introduce the terms in MAB in the context of AutoRAG-HP.

Arm In this context, an arm refers to a specific combination of hyper-parameters that we aim to optimize. For instance, if we are optimizing the top-k parameter, an arm can represent a candidate value for top-k (e.g., $\mathcal{K} = 3$). When optimizing multiple hyper-parameters simultaneously, an arm corresponds to a combination of these hyper-parameters, as defined in the standard formulation of the MAB problem. Note that since arms are discrete, the search space must first be discretized.

Trial A trial is a single iteration in which the algorithm selects an arm, observes the associated reward, and updates its estimation. In the RAG system, a trial can involve evaluating a group of queries with batch size B for the current selection of hyper-parameter combinations (i.e., arms). Optimal settings may be determined after a predetermined number of iterations, T, or upon meeting a specific stopping criterion.

Reward The reward function represents the user's objective and guides arm selection during the optimization process. For AutoRAG-HP, common goals include the maximization of response accuracy while paying less attention to the cost of LLM API calls (quantified by input token count), or balancing these objectives. For simplicity, we formulate the reward function as a linear combination of response accuracy and input token length:

$$\text{Reward} = w \cdot acc - (1 - w) \cdot \frac{t}{t_{max}}, \quad (1)$$

where w is the balance weight, t denotes the input token length and normalized by the maximal input token length t_{max}, and acc represents the LLM's response accuracy.

Optimization Algorithm Several optimization algorithms in MAB can guide arm selection based on the given reward function. One common choice is the Upper Confidence Bound (UCB) algorithm (Auer et al., 2002), which effectively balances exploration and exploitation by selecting arms based on their upper confidence bounds. These bounds are derived from confidence intervals

representing the estimated ranges of arm values. The UCB selection of arms is shown below:

$$A_t = \arg\max_{a \in \mathcal{A}} \left(Q_t(a) + \alpha \sqrt{\frac{\ln(t)}{N_a(t)}} \right), \quad (2)$$

where A_t is the selected arm, *i.e.*, the selected hyper-parameter or its combination, at timestep t, and $Q_t(a)$ is the estimated value of arm a at t. The square-root term quantifies the uncertainty. $N_a(t)$ represents the number of times arm a has been selected, and α is the hyper-parameter adjusting the balance between exploration and exploitation. During the iteration, both Q and the upper confidence bound (the square-root term) for each arm are updated to guide the selection of arms.

Thompson Sampling (TS) (Chapelle and Li, 2011) is another popular optimization algorithm in MAB. It balances exploration and exploitation by sampling from the posterior distribution of each arm's reward. Arms are chosen based on the highest sampled reward.

In summary, the objective of MAB is to maximize the total reward over a series of selections, even when the probability distribution of rewards for each arm is unknown. After a number of trials, the arm with the highest cumulative reward becomes the desired RAG hyper-parameter. This approach is particularly suitable for cold start problems, where prior estimation of user data is unavailable, and leveraging the MAB framework enables rapid tuning of hyper-parameters in RAG.

3.2 Two-level Hierarchical MAB

Applying the above standard formulation of MAB to hyper-parameter tuning in RAG can lead to the issue of having too many arms when jointly optimizing several hyper-parameters, resulting in an excessively large search space since it requires flattening the search space to obtain discrete arms. To mitigate this issue, we propose a two-level hierarchical MAB (Hier-MAB) where we first select which hyper-parameter to tune and then select one of its possible values.

In Figure 2, we show an example of two-level Hier-MAB in the context of jointly tuning of top-k ($\mathcal{K}$), embedding model ($\mathcal{E}$), and compression ratio ($\mathcal{C}$) hyper-parameters. The high-level arm is responsible for selecting which hyper-parameter to tune, while the lower-level arms control the hyper-parameter selection within the search space of each

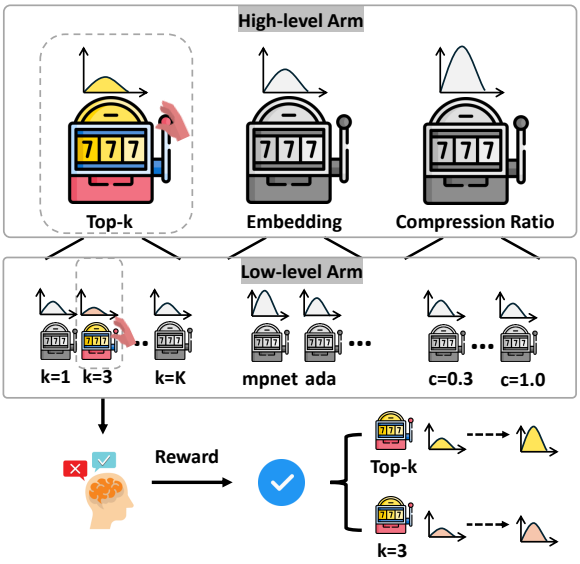

Figure 2: An example of two-level hierarchical MAB.

hyper-parameter. Thus, instead of having a single MAB, we now have four MABs: one for the high-level arm selection and the other three for the individual hyper-parameters. This hierarchical structure ensures that each MAB has a reasonable number of arms to select from, while all MABs combined can cover a large search space. This contrasts with the single MAB approach, which needs to enumerate all possible combinations when tuning multiple hyper-parameters.

The optimization process of Hier-MAB can be demonstrated by the trial shown in Figure 2. A high-level arm (top-k) is pulled, and within this hyper-parameter, the $\mathcal{K} = 3$ arm is pulled (with other hyper-parameters remaining the same as in the previous trial). After pulling the two-level arms and observing the associated reward, the algorithm updates its estimate of the selected arm's reward distribution using the new information, *i.e.*, updating the mean reward estimation and the confidence interval based on the observed reward. For the example in Figure 2, the positive reward updates the reward distribution of the pulled arms to reflect a higher estimated reward. Meanwhile, the reward distributions of other high- and low-level arms pulled in previous iterations also get updated. This process repeats for a predetermined number of iterations or until a stopping criterion is met.

4 Evaluation

4.1 Experiment Setup

Dataset We utilize the ALCE-ASQA (Gao et al., 2023) and Natural Questions (NQ)

datasets (Kwiatkowski et al., 2019) for our main experiments. Both datasets are in QA format and include candidate document chunks for each question. We use their evaluators to assess the accuracy of generated responses while adopting the prompts provided in Appendix F.

To accurately evaluate the capabilities of RAG systems in handling context-dependent queries, we exclude questions that can be correctly answered by LLMs without the need for external context, relying solely on their intrinsic knowledge. Although we also conducted experiments without this data filtering, which yielded comparable performance (see Appendix D for detailed results), the primary experimental results presented here are based on a sample of 350 questions from each benchmark after applying the data filtering process.

Note that to highlight the generalizability of our approach, we explored another RAG task, Text-to-SQL. More details on this can be found in Appendix A, due to space constraints in the main text.

Base LLMs We adopt GPT3.5-Turbo and GPT-4 models as base LLMs. Although the API parameters are tunable (Wang et al., 2023a), we opt to fix them by setting the temperature to zero and using default settings for all other parameters.

Search Space We examine the RAG setting as demonstrated in Figure 1. In the retrieval module, we evaluate the effects of the top-k hyper-parameter ($\mathcal{K}$) and the embedding model ($\mathcal{E}$) by considering three different choices, *i.e.*, "mpnet" (Song et al., 2020), "ada_002" (OpenAI, 2022), and "contriever" (Izacard et al., 2021). The compression module is implemented using the method outlined in the LLMLingua-2 work (Pan et al., 2024), with the compression ratio denoted as $\mathcal{C}$. Specifically, we consider two optimization tasks based on the number of hyper-parameters:

- Joint optimization of $(\mathcal{K}, \mathcal{C})$: They take discrete values from $\mathcal{K} \in [1, 3, 5, 7, 9]$ and $\mathcal{C} \in [0.3, 0.5, 0.7, 0.9, 1]$ while the embedding model is fixed to "mpnet".

- Joint optimization of $(\mathcal{K}, \mathcal{C}, \mathcal{E})$: We allow the embedding model to be tuned from the list of ["mpnet", "ada_002", "contriever"], maintaining the same settings for $\mathcal{K}$ and $\mathcal{C}$ as in the two-parameter case.

Reward Setting As outlined in the Methodology section, we introduce the weight parameter w to balance the tradeoff between token length and accuracy. Our experiments evaluate three values of w= 0.1, 0.5, and 0.9, corresponding to "cost-central", "balance", and "accuracy-central" regimes, respectively. The maximal token length t_{max} is set to be 1585 (2205) for ASQA (NQ) dataset. For better fit with MAB, we impose penalty for inaccurate response, *i.e.*, setting the accuracy acc to be -1.

Hier-MAB Setting We adopt UCB as the optimization algorithm for each arm selection in Hier-MAB, naming the approach as **Hier-UCB**. The parameter α for high-level and low-level arm selection with UCB are denoted as α^h and α^l, respectively, and are fixed at 1. To reduce sample variance during optimization, we use a batch size of $B = 4$.

Baseline Methods To compare with the proposed Hier-UCB approach, we evaluate three other online learning methods as follows:

- **UCB** (Auer, 2002): In this standard form, the search space is flattened out and a single UCB-based MAB is used for optimal hyper-parameter search. For consistency, α is also set to 1.

- **Thompson Sampling** (Agrawal and Goyal, 2013): TS samples arms from the posterior distribution of arms' rewards, with no pre-determined parameters.

- **Random Search** (Bergstra and Bengio, 2012b): This baseline selects arms uniformly at random, ensuring even exploration but without leveraging past rewards for guidance.

Ground-Truth and Evaluation Metric The ground-truth parameter combinations are determined using the **Grid Search** method (Lecun et al., 1998), which exhaustively evaluates all hyper-parameter combinations on the entire dataset. Grid Search serves as an offline benchmark against which the online learning methods are evaluated.

For the evaluation metric at a given timestamp t, we identify the top x hyper-parameter combinations from the evaluation method and calculate the percentage of these combinations that match the top x hyper-parameter combinations identified by Grid Search. Similar to metrics used in recommendation systems, we refer to this metric as **Recall**@x. Specifically, Recall@3 is used for the evaluation of the optimization of ($\mathcal{K}$, $\mathcal{C}$) and Recall@5 is used for the ($\mathcal{K}$, $\mathcal{C}$, $\mathcal{E}$) case. To mitigate statistical fluctuations, we conduct each experiment setting 10 times with different random seeds.

4.2 Experiment Result

Due to space constraints, we mainly present the experimental results for GPT-4 and leave the results for GPT-3.5-Turbo in the Appendix C. The following observations are similar for both cases.

Before diving into the evaluation results of various optimization methods, we first discuss the complexity of each optimization task.To illustrate this, we show the Grid Search results for ASQA dataset in Figure 3, presenting the accuracy and reward values across different weight settings for all three hyper-parameter combinations in search space. To show the sample variance during online learning, we plot the standard deviations of the accuracy and reward values across all batches as error bars.

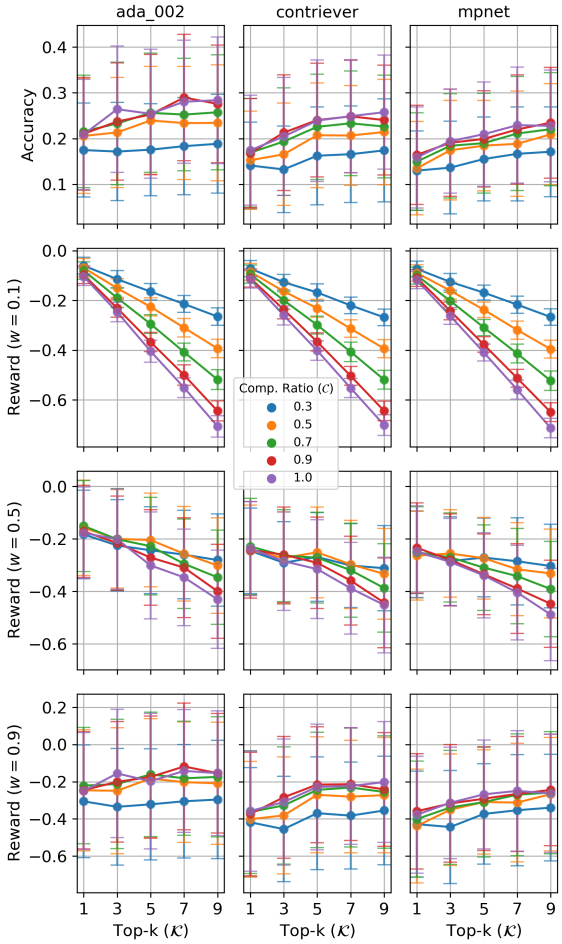

Figure 3: Grid search results for ASQA with GPT-4. Error bars represent the standard deviations of accuracy and reward values across all batches.

By inspecting Figure 3 and the other Grid Search results in Figure 11 (shown in Appendix B) for the NQ dataset, we make the following observations:

- The top-k and compression ratio have prominent impact on the response accuracy, while the effect of embedding model is more evident in ASQA as compared to NQ. This highlights the necessity of tuning those hyper-parameters, especially for the "accuracy-central" scenario (*i.e.*, $w = 0.9$).

- By factoring in more weights for token length, the overall landscape of reward function changes. For the "cost-central" scenario ($w = 0.1$), the preferred optimal settings would be small values of $\mathcal{K}$ and $\mathcal{C}$. Due to reduced dependence on response accuracy, sample variance becomes much smaller and the reward function over $(\mathcal{K}, \mathcal{C})$ exhibits steeper gradients. This indicates that the tuning over $(\mathcal{K}, \mathcal{C})$ when $w = 0.1$ can be relatively easy to achieve.

- With higher weights on accuracy ($w = 0.5$ and 0.9), reward function over $(\mathcal{K}, \mathcal{C})$ becomes less steep, accompanied by increased sample variance. This leads to many hyper-parameter combinations achieving similarly high rewards, a phenomenon known as search space degeneracy, which complicates the search for optimal settings. For instance, in the two-parameter case with $w = 0.9$, the panel in the last row and column of Figure 3 illustrates that the higher reward parameter region is relatively "flat", with clustered $(\mathcal{K}, \mathcal{C})$ combinations yielding similar rewards. Further tuning on the embedding model $\mathcal{E}$ in ASQA mitigates parameter degeneracy by enabling the distinction of $(\mathcal{K}, \mathcal{C})$ combinations with similar rewards. Conversely, in NQ, where the embedding model is less influential, adding it exacerbates the problem.

Based on the qualitative analysis, we catego-

w	Param.	Dataset	Complexity	Recall@x (All Avg.)	Recall@x (Hier-UCB)
0.1	$(\mathcal{K}, \mathcal{C})$	ASQA	Easy	0.83	0.76
		NQ	Easy	0.82	0.83
	$(\mathcal{K}, \mathcal{C}, \mathcal{E})$	ASQA	Easy	0.87	0.84
		NQ	Medium	0.68	**0.72**
0.5	$(\mathcal{K}, \mathcal{C})$	ASQA	Hard	0.20	0.10
		NQ	Hard	0.24	0.17
	$(\mathcal{K}, \mathcal{C}, \mathcal{E})$	ASQA	Medium	0.59	**0.64**
		NQ	Hard	0.25	0.32
0.9	$(\mathcal{K}, \mathcal{C})$	ASQA	Hard	0.31	0.30
		NQ	Hard	0.38	0.47
	$(\mathcal{K}, \mathcal{C}, \mathcal{E})$	ASQA	Medium	0.38	**0.6**
		NQ	Hard	0.27	0.3

Table 1: Complexity of optimization task for the GPT-4 case. The evaluation metrics Recall@3 when $T \times B = 2000$ and Recall@5 when $T \times B = 6000$ are used for $(\mathcal{K}, \mathcal{C})$ and $(\mathcal{K}, \mathcal{C}, \mathcal{E})$, respectively. The fifth column reports the metrics averaged over all methods, while the last column for the Hier-UCB only.

rize the optimization tasks by complexity ("Easy", "Medium" and "Hard"), as outlined in Table 1. Although the experiments are conducted on specific scenarios derived from various reward settings within two datasets, generalizing these scenarios by complexity levels provides insights into the broader applicability of the optimization method. In the following discussion, we will primarily reference the scenarios according to their complexity.

In Table 1 we also show the average evaluation result across all optimization methods (*i.e.*, Hier-UCB, UCB, TS and Random) as well as the result solely for Hier-UCB. The average Recall@x values (the 5th column) align well with our qualitative assessment of task complexity, *i.e.*, achieving ~ 0.8 for "Easy" tasks, ~ 0.5 for "Medium" tasks, and $\lesssim 0.3$ for "Hard" tasks. The iteration process concludes when $T \times B = 2000$ (for the 2-parameter case) or 6000 (for the 3-parameter case), with the number of LLM API calls being roughly 20% of those required for Grid Search.

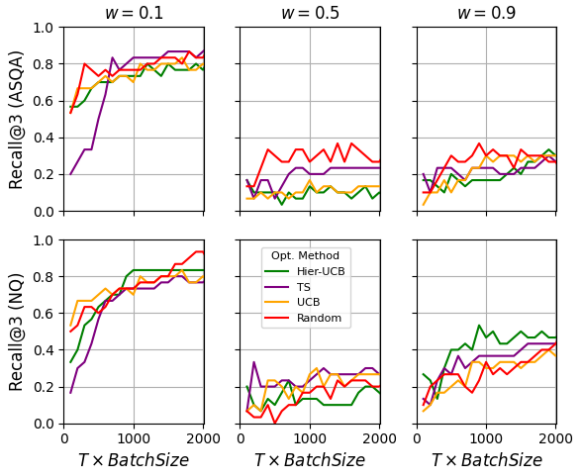

Figure 4: Evolution of Recall@3 in the optimization of $(\mathcal{K}, \mathcal{C})$ for the GPT-4 case.

Next, we compare Hier-UCB's performance with other baseline methods. In Figures 4 and 5, we plot the evolution of Recall@x metric over the iteration process for the 2-parameter and 3-parameter optimization cases, respectively. With the identification of task complexity in Table 1, the following observations are made:

- **Hier-UCB consistently outperforms other baselines for all "Medium" complexity cases**, while demonstrating comparable performance in "Easy" and "Hard" cases. Notably, Hier-UCB achieves faster convergence in "Medium" cases, as evident in Figure 5. The last column of Table 1 presents the Recall@x for Hier-UCB at the final timestamp, showing its competitive edge. However, this advantage is less pronounced compared to the mid-iteration timestamp (e.g., $T \times B \approx 2500$).

- All three baseline methods exhibit similar behavior. Although random exploration can be effective, its application in real-world online tuning requires caution. This approach is more likely to explore cases resulting in low rewards, thereby negatively impacting user experience.

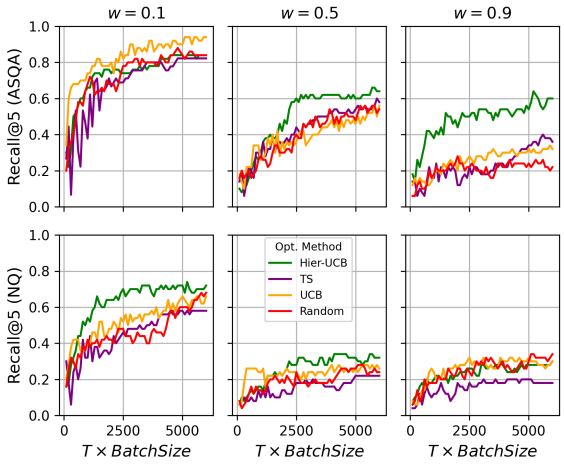

Figure 5: Evolution of Recall@5 in the optimization of $(\mathcal{K}, \mathcal{C}, \mathcal{E})$ for the GPT-4 case.

Lastly, we also observe that Hier-UCB achieves the highest average cumulative rewards over time, suggesting that it effectively balances exploration and exploitation by selecting hyperparameters that have minimal impact on user experience. Details can be found in Figure 15 in Appendix E.

4.3 Ablation Study

The results of Hier-UCB in the previous section are obtained with $\alpha^h = \alpha^l = 1$ and a batch size of $B = 4$. We now examine the impact of varying these values on performance. For this analysis, we focus on the 3-parameter optimization case, which encompasses all three complexity levels. Specifically, we present ablation studies for $\alpha^{h,l}$ in Figure 6 and for the batch size B in Figure 7.

From Figure 6, it can be observed that setting $\alpha^h = \alpha^l = 1$ is robust across all cases. Furthermore, a large α^l (e.g., $\alpha^l = 1.5$) can degrade performance in some cases, while setting a high value for α^h and a low value for α^l (e.g., $\alpha^h = 1.5, \alpha^l = 0.5$) yields the best overall per-

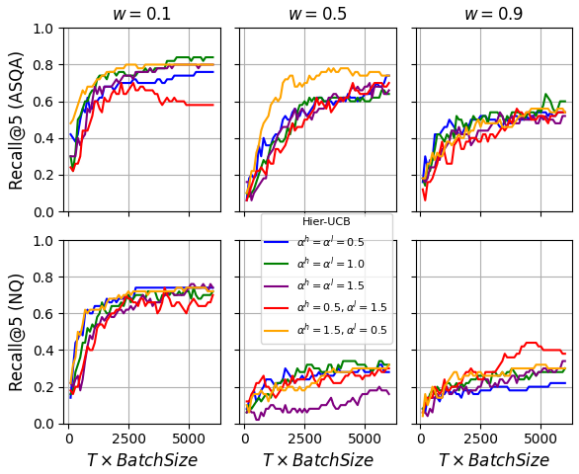

Figure 6: Evolution of Recall@5 when optimizing $(\mathcal{K}, \mathcal{C}, \mathcal{E})$ with varying $\alpha^{h,l}$ settings of Hier-UCB in the GPT-4 case.

formance. This can be understood intuitively: a higher value for the high-level arm, responsible for hyper-parameter selection, promotes exploration at the high level, avoiding premature convergence to local minima. Conversely, a lower α value in the lower-level arm facilitates faster convergence.

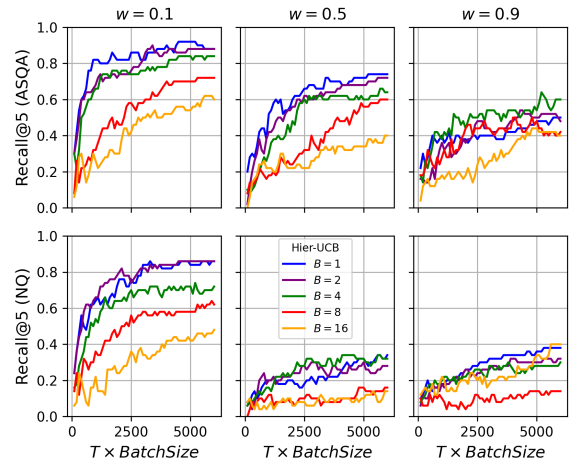

Figure 7: Evolution of Recall@5 when optimizing $(\mathcal{K}, \mathcal{C}, \mathcal{E})$ with varying batch sizes B of Hier-UCB in the ASQA GPT-4 case.

According to Figure 7, the batch size B greatly affects the performance. Again, $B = 4$ appears to be a robust choice across all cases. A smaller B may increase sample variance, particularly in the "accuracy-central" case ($w = 0.9$), and a larger B reduces the number of iterations, impairing the exploration process.

4.4 Case Study: Upgrade Base LLM from GPT-3.5-Turbo to GPT-4

Lastly, we demonstrate the application of our proposed Hier-UCB method in a real-world scenario: the upgrade of base LLMs. Given the rapid advancements in LLMs, there is a strong motivation to upgrade to a more advanced version for improved performance. In our experiment, we first conduct online hyper-parameter tuning with GPT-3.5-Turbo for $T \times B \in [0, 6000]$. At $T \times B = 6000$, we switch the base LLM to GPT-4. During this transition, we evaluate two configurations: **Continue** and **Reset**. The Continue configuration maintains the internal state parameters (e.g., $Q_t(a)$) from Eq. (2), effectively providing a warm start for later parameter search in GPT-4. In contrast, the Reset configuration initializes these parameters anew, simulating a cold start.

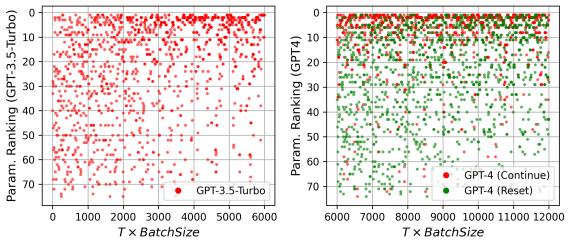

Figure 8: Evolution of an example three-parameter search in Hier-UCB when the base LLM is upgraded from GPT-3.5-Turbo (left) to GPT-4 (right), using the ASQA dataset, $w = 0.5$, $\alpha^h = \alpha^l = 1$, and $B = 1$. The Y-axis represents the ranking of parameter combinations for each base LLM taking from Grid Search, with lower values indicating higher rankings. Two different configurations of Continue and Reset are considered during the parameter search in GPT-4.

Figure 8 illustrates a three-parameter search trajectory when switching from GPT-3.5-Turbo to GPT-4, using the ASQA dataset. We set $w = 0.5$, $\alpha^h = \alpha^l = 1$, and $B = 1$. The Y-axis represents the ranking of parameter combinations for each base LLM taking from Grid Search, with lower values indicating higher rankings. In the first half of the process (left subplot), the Hier-UCB method effectively identifies optimal parameter combinations, evidenced by the clustering of parameters at the top in later timestamps. After transitioning to GPT-4, the Hier-UCB method quickly adapts under the Continue configuration, focusing on higher-ranked parameter combinations. However, under the Reset configuration, it explores more lower-ranked combinations, suggesting a need for addi-

tional exploration to find the optimal parameters.

To highlight the superior performance of the Continue configuration, Figure 9 presents the mean Recall@5 metric for 10 random trials. It reveals that the Continue configuration not only converges faster but also achieves significantly higher Recall@5 values. In summary, this experiment indicates that maintaining internal parameters during system changes can enhance the Hier-UCB method's adaptability and effectiveness.

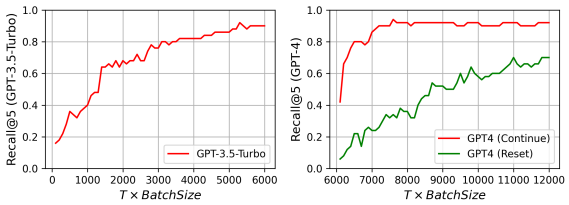

Figure 9: Evolution of the mean Recall@5 metric for 10 random trials, using the settings in Figure 8.

5 Discussion

In this work, we address online optimization in RAG systems by framing hyper-parameter tuning as an online Multi-Armed Bandit problem. Our proposed Hier-MAB approach can be extended to offline hyper-parameter tuning, demonstrating greater efficiency than traditional Grid Search methods, especially in scenarios with large search spaces with steep gradients. Hier-MAB can also serve as an initial step to filter the search space, followed by a more extensive parameter scan.

Our approach can be applied to a broader range of tunable hyper-parameters. While we focus on hyper-parameters in retrieval and prompt compression modules, it is extendable to other RAG modules such as document chunk size in indexing module, LLM API settings, or other LLM-based solutions (e.g., agent frameworks). Due to computational constraints, exhaustive searches for optimal configurations as the ground-truth are challenging, limiting the feasibility of experiments across broader hyper-parameter combinations.

The reward settings in our work can also be expanded. Currently, we assume the reward is a linear combination of accuracy and input token length, with a user-defined weight parameter allowing users to adjust the weight parameter w. However, determining the right parameter to balance accuracy and LLM API cost is challenging in reality. An alternative is to set LLM API cost constraints and optimize accuracy within these constraints, incorporating cost constraints as penalty terms in the reward function. This converts a multi-objective optimization problem into a single-objective one, though exploring Pareto optimization within RAG could also yield valuable insights.

Our current reward framework only considers feedback from the correctness of the final response. In practice, feedback might also come from intermediate steps (e.g., document relevance evaluation in retrieval modules) or multiple sources in multi-turn dialogues. Thus, automatically and efficiently integrating these additional feedback sources into the reward definition is also worth exploration.

Beyond hyper-parameter tuning, developing a comprehensive AutoML framework for RAG involves identifying the optimal combination of available RAG modules, automated prompt tuning and other query-dependent parameters, such as those in a routing module that directs queries to appropriate base LLMs. Additionally, an ideal AutoRAG system should auto-generate evaluation data for tuning as needed, supporting the "Bring Your Data" vision where users provide their data, and the platform autonomously configures the entire pipeline to meet their specific requirements. Future work will explore these areas.

6 Summary

Inspired by traditional AutoML practices designed to simplify and automate ML/AI development, we introduce the AutoRAG-HP framework. This framework addresses the critical need for efficient and effortless hyper-parameter tuning within the Retrieval-Augmented Generation (RAG) system in the context of LLMs. To address challenges posed by extensive search spaces and the need for online tuning, we formulate hyper-parameter selection in RAG as a multi-armed bandit problem and introduce a novel two-level hierarchical Upper Confidence Bound (Hier-UCB) method for efficient parameter space exploration.

Our experiments on simultaneously tuning three hyper-parameters demonstrate that multi-armed bandit-based online learning methods (Hier-UCB, UCB, and TS) can achieve Recall@5 ≈ 0.8 for scenarios with prominent gradients in search space, using only $\sim 20\%$ of the LLM API calls required by the Grid Search approach. Additionally, the proposed Hier-UCB approach outperforms other baselines in more challenging optimization scenarios. These promising results motivate further exploration into automatic tuning of the RAG system to achieve the full vision of AutoRAG.

Limitations

We acknowledge the limitations of this paper. First, we evaluate AutoRAG-HP using only two LLMs as backbones. Additional experiments can be done to assess AutoRAG-HP's performance with other LLMs as well as small language models. Secondly, our experiments are limited to two public datasets in QA format. Further testing can be done across diverse tasks and datasets. Finally, we only explore jointly tuning of up to three hyper-parameters and further exploration can be extended to include tuning a greater number of hyper-parameters, which we will leave for future work.

Ethics Statement

This paper focuses on hyper-parameter optimization and does not inherently address potential risks associated with the underlying LLMs, such as unethical outputs, toxicity, and biases. We strongly recommend integrating Responsible AI modules within the RAG pipeline and conducting a comprehensive evaluation of these potential issues prior to deployment in practice.

Acknowledgments

We would like to thank Henry Zeng and Victor Rühle for insightful discussion on building efficient RAG solutions. We are also indebted to Qianhui Wu, Huiqiang Jiang and Bo Qiao for their help in establishing the prompt compression API.

References

Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *International conference on machine learning*, pages 127–135. PMLR.

Peter Auer. 2002. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422.

Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Machine learning*, 47:235–256.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011a. Algorithms for hyper-parameter optimization. In *Neural Information Processing Systems*.

James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011b. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA. Curran Associates Inc.

James Bergstra and Yoshua Bengio. 2012a. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.*, 13(null):281–305.

James Bergstra and Yoshua Bengio. 2012b. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(2).

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Shuaichen Chang and Eric Fosler-Lussier. 2023. Selective demonstrations for cross-domain text-to-SQL. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14174–14189, Singapore. Association for Computational Linguistics.

Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. *Advances in neural information processing systems*, 24.

Dujian Ding, Ankur Mallick, Chi Wang, Robert Sim, Subhabrata Mukherjee, Victor Ruehle, Laks V. S. Lakshmanan, and Ahmed Awadallah. 2024. Hybrid llm: Cost-efficient and quality-aware query routing. In *ICLR 2024*.

Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. 2015. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. 2023. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6465–6488.

Frank Hutter, Lars Kotthoff, and Joaquin Vanschoren. 2019. Automated machine learning: Methods, systems, challenges. *Automated Machine Learning*.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2021. Unsupervised dense information retrieval with contrastive learning.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard

Grave. 2022. Atlas: Few-shot learning with retrieval augmented language models.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023a. Llmlingua: Compressing prompts for accelerated inference of large language models. *Preprint*, arXiv:2310.05736.

Zhengbao Jiang, Frank F. Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023b. Active retrieval augmented generation.

Haifeng Jin, François Chollet, Qingquan Song, and Xia Hu. 2023. Autokeras: An automl library for deep learning. *Journal of Machine Learning Research*, 24(6):1–6.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Tze Leung Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Advances in applied mathematics*, 6(1):4–22.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

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. Curran Associates, Inc.

Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670.

Xueqing Liu and Chi Wang. 2021. An empirical study on hyperparameter optimization for fine-tuning pre-trained language models. In *Annual Meeting of the Association for Computational Linguistics*.

Yuanjie Lyu, Zhiyu Li, Simin Niu, Feiyu Xiong, Bo Tang, Wenjin Wang, Hao Wu, Huanyong Liu, Tong Xu, Enhong Chen, Yi Luo, Peng Cheng, Haiying Deng, Zhonghao Wang, and Zijia Lu. 2024. Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models. *Preprint*, arXiv:2401.17043.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. 2023. Query rewriting for retrieval-augmented large language models.

Marker-Inc-Korea. 2024. https://github.com/marker-inc-korea/autorag.

Randal S. Olson, Nathan Bartley, Ryan J. Urbanowicz, and Jason H. Moore. 2016. Evaluation of a tree-based pipeline optimization tool for automating data science. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, GECCO '16, pages 485–492, New York, NY, USA. ACM.

OpenAI. 2022. https://openai.com/index/new-and-improved-embedding-model/.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, Dongmei Zhang, Karl Cobbe, Vineet Kosaraju, Mo Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, and Reiichiro Nakano. 2024. Llmlingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. *ArXiv*, abs/2403.12968.

Reid Pryzant, Dan Iter, Jerry Li, Yin Lee, Chenguang Zhu, and Michael Zeng. 2023. Automatic prompt optimization with "gradient descent" and beam search. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 7957–7968, Singapore. Association for Computational Linguistics.

Anthony Sarah, Sharath Nittur Sridhar, Maciej Szankin, and Sairam Sundaresan. 2024. Llama-nas: Efficient neural architecture search for large language models.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding.

Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, and Marius Lindauer. 2024. AutoML in the age of large language models: Current challenges, future opportunities and risks. *Transactions on Machine Learning Research*.

Marcos Treviso, Ji-Ung Lee, Tianchu Ji, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Colin Raffel, Pedro H. Martins, André F. T. Martins, Jessica Zosa

Forde, Peter Milder, Edwin Simpson, Noam Slonim, Jesse Dodge, Emma Strubell, Niranjan Balasubramanian, Leon Derczynski, Iryna Gurevych, and Roy Schwartz. 2022. Efficient methods for natural language processing: A survey.

Joannes Vermorel and Mehryar Mohri. 2005. Multi-armed bandit algorithms and empirical evaluation. In *European conference on machine learning*, pages 437–448. Springer.

Chi Wang, Susan Xueqing Liu, and Ahmed H. Awadallah. 2023a. Cost-effective hyperparameter optimization for large language model generation inference.

Chi Wang, Qingyun Wu, Silu Huang, and Amin Saied. 2021. ECONOMIC HYPERPARAMETER OPTIMIZATION WITH BLENDED SEARCH STRATEGY. In *International Conference on Learning Representations*.

Chi Wang, Qingyun Wu, Markus Weimer, and Erkang Zhu. 2019. Flaml: A fast and lightweight automl library.

Liang Wang, Nan Yang, and Furu Wei. 2023b. Query2doc: Query expansion with large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9414–9423, Singapore. Association for Computational Linguistics.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium. Association for Computational Linguistics.

Ruiqi Zhong, Tao Yu, and Dan Klein. 2020. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 396–411, Online. Association for Computational Linguistics.

Lucas Zimmer, Marius Lindauer, and Frank Hutter. 2021. Auto-pytorch: Multi-fidelity metalearning for efficient and robust autodl. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(9):3079–3090.

A Result for Text-to-SQL Task

Here, we present the experimental results by tuning four hyperparameters in a new RAG task, specifically a Text-to-SQL task where SQL queries are generated from users' natural language queries. Our experiments employed the ODIS method (Chang and Fosler-Lussier, 2023), which utilizes few-shot examples from both in-domain and out-of-domain databases for prompt construction. Following the ODIS framework, we considered four tunable hyperparameters: the in-domain retrieval strategy, the number of out-of-domain databases, the number of out-of-domain examples, and the number of synthetic in-domain examples. For in-domain retrieval, we tested three settings ('similarsql', 'simsql_pred', 'covsql'), while each of the other three parameters had four candidate values ([2, 3, 4, 5]). This resulted in 192 combinations of hyperparameter settings.

We randomly selected 100 test samples from the Spider dataset (Yu et al., 2018) and evaluated the accuracy of the generated SQL queries by executing them on the corresponding databases using the test-suite-sql-eval repository (Zhong et al., 2020). We performed experiments with Hier-UCB and three baselines (Random, TS, and UCB), using the same settings as the main text. The Recall@10 metrics at $T \times B = 4000$ ($\sim 20\%$ of the computation cost of the Grid Search method) are presented in Table 2. Hier-UCB outperformed the other baselines for weights of 0.5 and 0.9 but lagged behind UCB for a weight of 0.1. The results for $w = 0.1$ were expected since this scenario emphasizes token length, which is simpler than the others. As noted in the main text, Hier-UCB performs similarly to other baselines in less complex scenarios.

In summary, this experiment on a different RAG task verifies the effectiveness of our proposed Hier-UCB method across diverse scenarios and a broader range of RAG parameters.

B Grid Search Results for ASQA and NQ

Grid Search results for the ASQA dataset with GPT-3.5-Turbo is shown in Figures 10. Grid Search results for the NQ dataset with GPT-4 and GPT-3.5-Turbo are shown in Figures 11 and Figures 12, respectively.

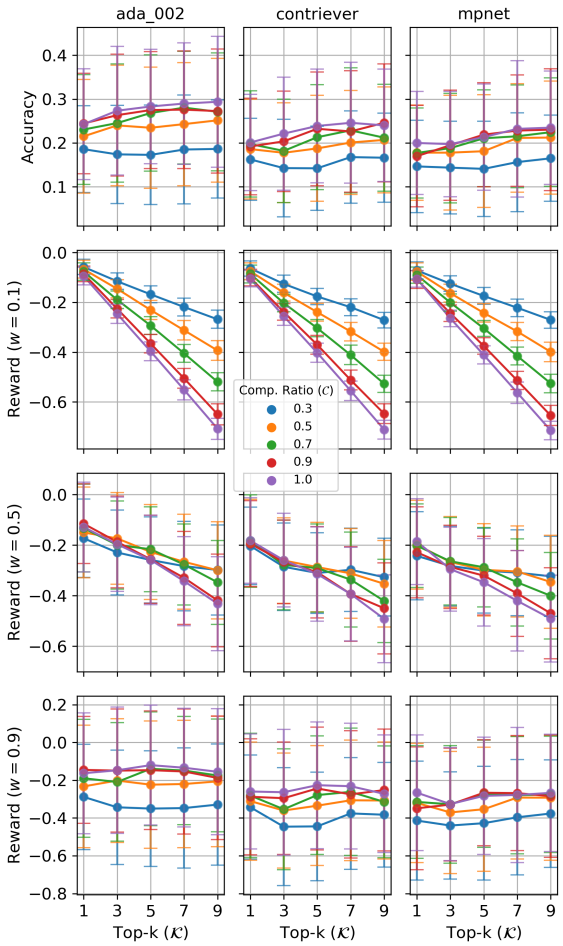

Figure 10: Grid search results for ASQA with GPT-3.5-Turbo. Error bars represent the standard deviations of accuracy and reward values across all batches.

C Result for ASQA and NQ using GPT-3.5-Turbo

In Figures 13 and 14, we plot the the experimental results with GPT-3.5-Turbo, *i.e.*, evolution of Recall@x metric over the iteration process for the 2-parameter and 3-parameter optimization cases, respectively.

D Result without Question Filtering

We conducted experiments in the NQ dataset without filtering out the questions that can be answered correctly by LLMs without context. The results for the 3-parameter case with $T \times B = 6000$ and GPT-4 are provided in Table 3. Similar outcomes were observed across other settings. Hier-UCB consistently outperforms the baselines in medium-complexity scenarios (weight = 0.9) and shows similar performance in both easy (weight = 0.1) and highly complex scenarios (weight = 0.5).

Weight	Recall@10 (Hier-UCB)	Recall@10 (Random)	Recall@10 (TS)	Recall@10 (UCB)	Recall@10 (all avg.)
0.1	0.40	0.03	0.35	**0.49**	0.29
0.5	**0.35**	0.05	0.23	0.26	0.18
0.9	**0.21**	0.07	0.08	0.11	0.08

Table 2: Evaluation results for Text-to-SQL task when $T \times B = 4000$ using GPT-4.

Weight	Recall@5 (Hier-UCB)	Recall@5 (TS)	Recall@5 (Random)	Recall@5 (UCB)
0.1	**0.74**	0.60	0.68	0.70
0.5	0.04	0.06	0.08	**0.14**
0.9	**0.6**	0.36	0.22	0.22

Table 3: Evaluation results without data filtering in NQ for 3-parameter case $(\mathcal{K}, \mathcal{C}, \mathcal{E})$ when $T \times B = 6000$ and using GPT-4.

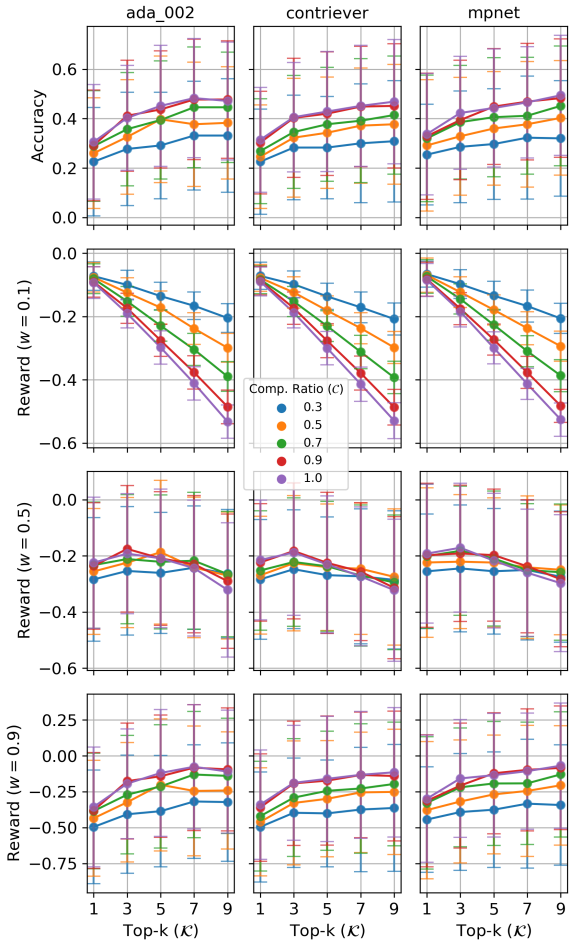

Figure 11: Grid search results for NQ with GPT-4. Error bars represent the standard deviations of accuracy and reward values across all batches.

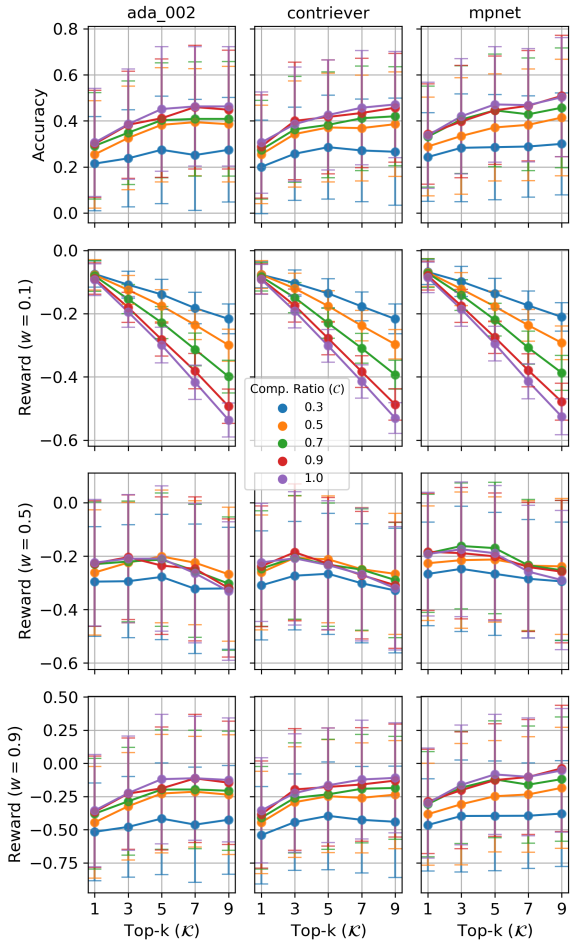

Figure 12: Grid search results for NQ with GPT-3.5-Turbo. Error bars represent the standard deviations of accuracy and reward values across all batches.

E Trade-offs Between Exploration and Exploitation

We highlight the critical trade-off between exploration and exploitation in online learning. Below, we present the average cumulative rewards at different timesteps ($T \times B$) for Hier-UCB and other baselines for the ASQA dataset using GPT-4 with a weight of 0.5. The results are consistent across other weight settings and the NQ dataset. As shown in Figure 15, it indicates that Hier-UCB achieves the highest average cumulative rewards over time,

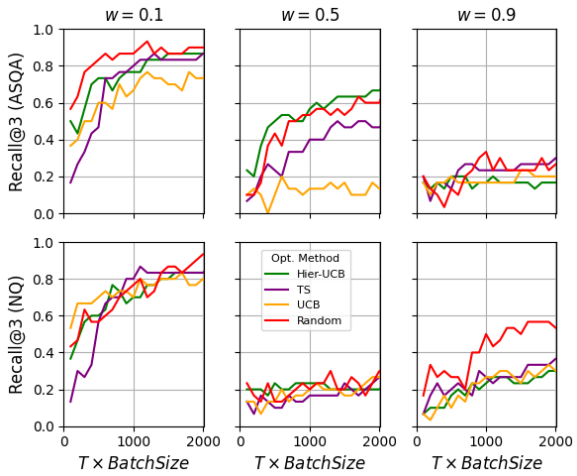

Figure 13: Evolution of Recall@3 in the optimization of $(\mathcal{K}, \mathcal{C})$ for the GPT-3.5-Turbo case.

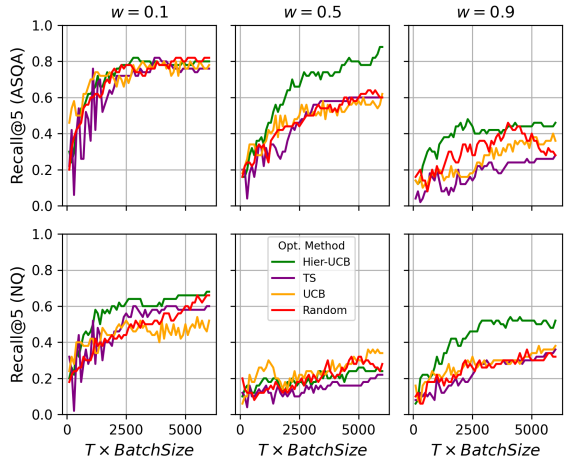

Figure 14: Evolution of Recall@5 in the optimization of $(\mathcal{K}, \mathcal{C}, \mathcal{E})$ for the GPT-3.5-Turbo case.

suggesting that it effectively balances exploration and exploitation by selecting hyperparameters that have minimal impact on user experience.

F Prompts

Examples of prompts for the evaluation of ASQA and NQ datasets are in Tables 4 and 5 respectively. The examples shown here are with the ($\mathcal{K} = 3$ and $\mathcal{C} = 1$) setting.

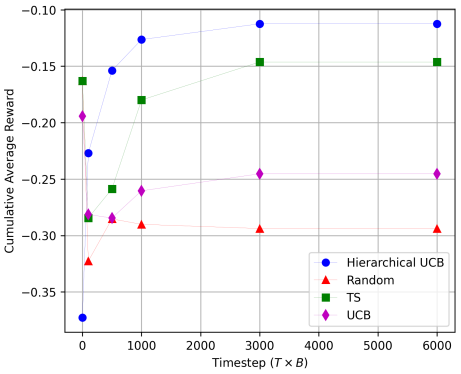

Figure 15: Comparison of cumulative rewards among different algorithms at different timesteps of $T \times B$.

Instruction: Write an accurate, engaging, and concise answer for the given question using only the provided search results (some of which might be irrelevant) and cite them properly. Use an unbiased and journalistic tone. Always cite for any factual claim. When citing several search results, use [1][2][3]. Cite at least one document and at most three documents in each sentence. If multiple documents support the sentence, only cite a minimum sufficient subset of the documents.

Question: Who has the highest goals in world football?

Document [1](Title: Argentina–Brazil football rivalry): "Football Player of the Century", by IFFHS International Federation of Football History and Statistics, 1999, "South America Football Player of the Century", by IFFHS International Federation of Football History and Statistics. Pelé's 1281 goals are recognized by FIFA as the highest total achieved by a professional footballer, although the Soccer Statistic Foundation (rssf) recognizes only 767 goals in official mode, occupying the third place after Josef Bican (805) and Romario (772). For his part, Maradona has been named the best soccer player in World Cup history both by The Times and FourFourTwo, publication that also rewarded him as the "Best

Document [2](Title: Godfrey Chitalu): have beaten Gerd Müller's record of 85 goals in a year, the Football Association of Zambia claimed that the world record actually pertained to Godfrey Chitalu who had scored 116 goals (possibly 117) during the 1972 calendar year and 107 during the 1972 season. The difference of goals is due to first 9 goals being scored before the season officially started. The Football Association of Zambia presented the evidence to FIFA but a spokesperson responded that they would ratify neither Lionel Messi's nor Chitalu's records as they do not keep statistical track of domestic competitions. Nonetheless, it could constitute the

Document [3](Title: Godfrey Chitalu): highest official tally claimed by a national football association. Chitalu made his international debut on 29 June 1968 in a friendly match against Uganda in Lusaka which Zambia won 2–1. He scored his first goal in a 2–2 draw against the same team five days later. Chitalu played a prominent role during the World Cup qualification matches against Sudan with Zambia being eliminated on a strange rule which was peculiar to Africa and favoured the team that won the second leg. Despite the aggregate score being tied at 6–6 after Zambia won the first leg 4–2 and lost the return

Answer:

Table 4: Prompt for ASQA. The prompt consists of Instruction, Question, and $\mathcal{K}$ retrieved Documents, where $\mathcal{K}$ in the table example is equal to 3 and without prompt compression.

Instruction: Write a high-quality answer for the given question using only the provided search results (some of which might be irrelevant).

Question: which is the default file extension for an audio file in windows media player

Document [1](Title: Windows Media Player) Windows Media Player 11 is available for Windows XP and included in Windows Vista and Windows Server 2008. The default file formats are Windows Media Video (WMV), Windows Media Audio (WMA), and Advanced Systems Format (ASF), and its own XML based playlist format called Windows Playlist (WPL). The player is also able to utilize a digital rights management service in the form of Windows Media DRM.

Document [2](Title: Windows Media Player) as data discs with playlists such as an MP3 CD, synchronize content with a digital audio player (MP3 player) or other mobile devices, and enable users to purchase or rent music from a number of online music stores. Windows Media Player replaced an earlier application called Media Player, adding features beyond simple video or audio playback. Windows Media Player 11 is available for Windows XP and included in Windows Vista and Windows Server 2008. The default file formats are Windows Media Video (WMV), Windows Media Audio (WMA), and Advanced Systems Format (ASF), and its own XML based playlist format called

Document [3](Title: Windows Media Audio) Windows Media DRM cannot play DRM-protected files. Windows Media Audio Windows Media Audio (WMA) is the name of a series of audio codecs and their corresponding audio coding formats developed by Microsoft. It is a proprietary technology that forms part of the Windows Media framework. WMA consists of four distinct codecs. The original WMA codec, known simply as "WMA", was conceived as a competitor to the popular MP3 and RealAudio codecs. "WMA Pro", a newer and more advanced codec, supports multichannel and high resolution audio. A lossless codec, "WMA Lossless", compresses audio data without loss of audio fidelity

Answer:

Table 5: Prompt for Natural Question. The prompt consists of Instruction, Question, and $\mathcal{K}$ retrieved Documents, where $\mathcal{K}$ in the table example is equal to 3 and without prompt compression.