Learning to Rank for Multiple Retrieval-Augmented Models through Iterative Utility Maximization

Alireza Salemi
University of Massachusetts Amherst
Amherst, MA, United States
asalemi@cs.umass.edu

Hamed Zamani
University of Massachusetts Amherst
Amherst, MA, United States
zamani@cs.umass.edu

Abstract

This paper investigates the design of a unified search engine to serve multiple retrieval-augmented generation (RAG) agents, each with a distinct task, backbone large language model (LLM), and RAG strategy. We introduce an iterative approach where the search engine generates retrieval results for the RAG agents and gathers feedback on the quality of the retrieved documents during an offline phase. This feedback is then used to iteratively optimize the search engine using an expectation-maximization algorithm, with the goal of maximizing each agent's utility function. Additionally, we adapt this to an online setting, allowing the search engine to refine its behavior based on real-time individual agents feedback to better serve the results for each of them. Experiments on datasets from the Knowledge-Intensive Language Tasks (KILT) benchmark demonstrates that our approach significantly on average outperforms baselines across 18 RAG models. We demonstrate that our method effectively "personalizes" the retrieval for each RAG agent based on the collected feedback. Finally, we provide a comprehensive ablation study to explore various aspects of our method.

CCS Concepts

• **Information systems** → **Learning to rank**; **Personalization**;
• **Computing methodologies** → **Natural language generation**.

Keywords

Retrieval-augmented generation; retrieval-enhanced machine learning; search engine for agents; ranking optimization

ACM Reference Format:
Alireza Salemi and Hamed Zamani. 2025. Learning to Rank for Multiple Retrieval-Augmented Models through Iterative Utility Maximization. In *Proceedings of the 2025 International ACM SIGIR Conference on Innovative Concepts and Theories in Information Retrieval (ICTIR) (ICTIR '25), July 18, 2025, Padua, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3731120.3744584

1 Introduction

Search engines have been designed mainly to serve people by providing relevant results for search queries. They are typically trained at scale using learning-to-rank methods with implicit feedback gathered and refined over time through user interactions [11, 17, 18]. These systems often use personalization to deliver more tailored and effective search results for users [5, 48]. With the recent rise of large language models (LLMs), along with their user-friendly chat interfaces, many users now use them for a wide range of tasks [4]. Extensive studies have shown that LLMs often face challenges in tasks that require external knowledge in rapidly changing or evolving environments [32, 62]. One method to address this limitation is to enhance LLMs by retrieving information from a knowledge source, a technique known as retrieval-augmented generation (RAG) [3, 15, 27]. This marks a paradigm shift in the current landscape, where humans interact primarily with LLMs, while LLMs rely on search engines to retrieve external information [21, 45, 62].

With this paradigm shift, it is crucial to develop methods for optimizing and personalizing search engines to better serve their new users: the LLMs that depend on them. This need is further highlighted by prior work showing significant differences between the information preferences of humans and RAG agents [44]. In addition, different LLMs may have different information needs. For example, smaller LMs with limited reasoning capabilities might prefer supporting documents that provide explicit information, while more powerful LLMs can infer details even when the information is stated vaguely. Given these variations, optimizing engines to address the unique requirements of different LLMs is essential.

Prior work on RAG mostly focus on developing a RAG system (i.e., a retrieval and a language model) for each task. The recent work by Salemi and Zamani [45] represents one of the first efforts to build a centralized search engine capable of serving *multiple* RAG agents, each tasked with a distinct objective. These agents utilize different underlying LLMs and employ varying RAG techniques. To train this centralized search engine, they use its initial parameters to retrieve documents for the training queries of the agents (i.e., the LLMs utilizing the search engine). The feedback from these agents on the retrieved documents is then collected to update the parameters. This approach has a few limitations. First, it depends heavily on the initial parameters to retrieve documents for training queries. If these parameters are not well-initialized, the quality of the initial retrieval lists would be low, leading to suboptimal feedback from the LLMs. Furthermore, since the initial documents are retrieved without any prior feedback to adapt the search engine to agent needs, they may not be relevant or useful to the agents. Consequently, the feedback provided might not reflect useful documents for the individual agents, reducing the overall effectiveness of the training process.

This paper addresses these limitations by introducing an iterative approach grounded in strong theoretical foundations, designed to maximize the utility functions of RAG agents. Feedback is collected over multiple iterations in an offline training procedure to progressively optimize the search engine. In each iteration, the

ICTIR '25, Padua, Italy

ACM ISBN 979-8-4007-1861-8/2025/07
https://doi.org/10.1145/3731120.3744584

search engine applies the parameters optimized from the previous feedback round to retrieve new documents tailored to the specific information needs of each RAG agent. Feedback on these newly retrieved documents is then collected to refine the search engine's understanding of each agent's preferences. The personalization process adapts the search engine for each RAG agent by leveraging feedback from the prior iteration, using the agents' training queries and feedback on them. During the offline phase, feedback from all agents is aggregated to train a multi-task central search engine capable of effectively serving the diverse needs of all agents.

A natural extension to the offline optimization phase is online learning in the serving phase where the retriever provides search results for the agents on test queries. Here, the same iterative algorithm is applied to adapt the retriever based on agent feedback obtained during active serving sessions. Specifically, the retriever first processes a batch of queries from an agent, collects feedback on these queries, and then updates its parameters to improve performance on subsequent queries within the same session for this specific agent. This optimization occurs concurrently with the serving process and is not confined to pre-defined training queries. Since this optimization leverages each agent's feedback individually to further optimize itself, it enhances the personalization of the search engine for that specific agent even more, tailoring the system to better meet the unique information needs of each agent.

We evaluate our approach using diverse tasks from the Knowledge-Intensive Language Tasks (KILT) benchmark [32]. Our evaluation includes three open-domain question answering datasets: Natural Questions (NQ) [24], TriviaQA [19], and HotPotQA [60]; one fact verification dataset: FEVER [53]; and two relation extraction and slot-filling datasets: zsRE [25] and T-REx [9]. Following Salemi and Zamani [45], we test our approach with 18 different RAG agents as the users, each performing a specific task and utilizing distinct augmentation approaches and LLMs, to serve as users of the search engine. Our results demonstrate that the proposed approach for offline iterative training of the search engine significantly outperforms the state-of-the-art baseline. Additionally, combining this offline approach with our online learning yields an even greater improvement over the baselines. We also conduct an extensive ablation study on various configurations of the proposed approaches to provide further insights into their effectiveness and impact. Furthermore, we show that our approach enhances the personalization of the search engine over time, addressing a limitation noted in Salemi and Zamani [45]. We observe that the correlation between the retrieval results of the agents employing different LLMs but performing the same task is very low, indicating that the results are effectively personalized. To support future research, we have open-sourced our code and models for the community. [1]

2 Related Work

Knowledge-Intensive Language Tasks (KILT). Contrary to standard NLP tasks like natural language understanding [55, 56] and question answering [29], where the input alone is enough to perform the task, knowledge-intensive NLP tasks rely heavily on external knowledge sources to extract necessary information. Petroni et al. [32] introduces KILT, a benchmark designed for evaluating such tasks. KILT encompasses a variety of tasks, including open-domain question answering, fact verification, slot filling, and entity linking, providing a benchmark for knowledge-intensive tasks.

Retrieval-Augmented Generation (RAG). RAG [27] represents a framework that merges information retrieval with natural language generation to improve the quality of generated outputs by integrating external knowledge in the generation process [3, 51]. Unlike traditional LLMs that rely solely on pre-trained knowledge, RAG can dynamically retrieve information from external sources via a retriever, enabling them to produce content that is more accurate [21, 62]. This flexibility allows RAG to be applied in various domains, including knowledge grounding in textual [15, 27, 32] and multimodal [6, 10, 37, 42], personalization [23, 38–41, 43], and reducing hallucinations [2, 47]. The retriever in RAG plays a pivotal role as it sources the necessary information for the LLM to perform its task [27]. This is typically done using either sparse retrieval methods (e.g., TF-IDF, BM25 [35]) or dense retrieval models (e.g., DPR [20], Contriever [13], ColBERTv2 [46], E5 [58]). The retrieved information is then utilized by the large language model to complete the task. Prominent methods in this context include In-Prompt Augmentation (IPA) and Fusion-in-Decoder (FiD) [15]. In IPA, the retrieved data is appended to the prompt, allowing the language model to incorporate it during generation. FiD encodes each retrieved document separately alongside the prompt within the encoder of an encoder-decoder architecture, combining them in the decoder to generate a cohesive answer based on the available information, as explained in Izacard and Grave [15].

A Search Engine for Machines. Research on search engines show that successful systems rely on large-scale feedback for optimization [1, 54]. With LLMs as the primary users of search engines [21, 62], Salemi and Zamani [44] showed that the LLMs' preferences about relevance of a query and document differs from humans. Salemi and Zamani [45] introduced a new problem that is training a unified search engine capable of serving multiple diverse RAG agents. They introduced uRAG, a unified ranking model designed to serve multiple RAG models while learning and optimizing based on feedback provided by these diverse RAG models. Recently, several methods have been proposed for training retrieval models tailored to LLMs, including distillation from LLMs to retrievers [14, 16, 59], end-to-end training of retrievers and LLMs [36, 61], and bandit algorithms [57]. However, most of these approaches focus on leveraging feedback from a single LLM, with the aim of aligning the retrieval with that particular LLM [45]. Here, we introduce an approach based on iterative utility maximization to train a unified retrieval model for serving multiple RAG agents, applied in both offline and online settings [7, 12] to optimize the search engine for the agents.

3 Problem Formulation

Consider the retrieval model R_θ, parameterized by θ, whose main role is to facilitate information access from a corpus C for a set of RAG models (a.k.a, RAG agents) denoted as $M = \{M_i\}_{i=1}^{n}$. Each M_i acts as a black-box agent for R_θ, which means that R_θ does not have access to the models' architecture, configuration, or parameters. Each M_i is designed to perform a knowledge-intensive task T_i =

[1] Our code can be found at: https://github.com/alirezasalemi7/uRAG

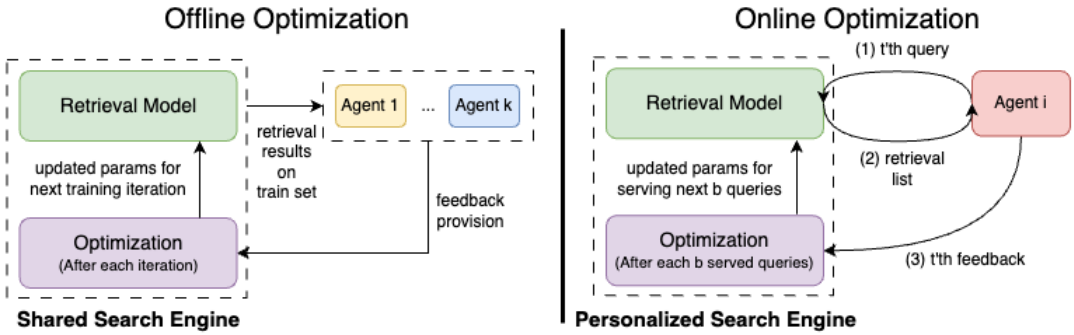


Figure 1: Iterative Utility Maximization (IUM). This framework first iteratively trains the search engine with feedback from all agents offline, then individually iteratively trains and serves the model for each agent during the online phase.

$(D_i^{\text{train}}, D_i^{\text{test}}, \mu_i)$ that requires external information from the corpus C as the knowledge source. There is a training dataset $D_i^{\text{train}} = \{(x_j, y_j)\}_{j=1}^{|D_i^{\text{train}}|}$ for each agent M_i, which can be used by the retrieval model R_θ for offline optimization. At inference, each agent M_i operates *sequentially* on a test dataset $D_i^{\text{test}} = \{(x'_j, y'_j)\}_{j=1}^{|D_i^{\text{test}}|}$ in the same order. The end-to-end performance of the agent M_i can be measured by a utility function (metric) μ_i.

Each RAG agent can be simply formulated as $\bar{y}_{M_i} = M_i(x; R_\theta)$. In more detail, given an input x, each RAG agent M_i submits a query to the retrieval model R_θ for information access, and generates the output by consuming the top k retrieved documents (i.e., $\mathcal{R}_k = \{r_1, ..., r_k\}$). As suggested in [45], the search engine and RAG agents can engage in an offline optimization process, in which each RAG agent M_i produces a feedback list for a retrieval list of size k, denoted as $f_{M_i} = \{f_j\}_{j=1}^{k} \in \{0, 1\}^{1\times k}$. This feedback list indicates the usefulness of each retrieved document in the retrieval list from the agent's perspective. Feedback can be computed in various ways; based on the performance of the generated output when utilizing the retrieved documents on the downstream task, e.g., using the utility function μ_i, or based on ratings received from the users or evaluators of the RAG agent, among other methods. Without loss of generality, this paper focuses on the first case as the main method for providing feedback from the agents. In this approach, feedback is based on the agent's downstream performance when it uses each document individually in its task. Therefore, the main goal of this paper is to optimize the retrieval model R_θ to maximize the the expected utility for all the RAG agents that use the retrieval model.

4 Learning to Rank with Iterative Utility Maximization

State-of-the-art commercial search engines have been trained using implicit feedback from their users, such as clicks, scrolling behavior, and dwell time [11, 17, 18]. In our setting, however, the primary users of the search engine are the RAG agents that consume the retrieval results. Therefore, their feedback on the quality of the retrieval results serves as the primary signal for optimizing the search engine. Previous work [45] has explored the use of this feedback to train the model, where they employed the initial parameters of the search engine—without any prior training—to retrieve a set of documents and collect feedback for them in an offline setting to optimize the search engine. This approach has several shortcomings. First, using an untrained retrieval model to gather documents for feedback collection is suboptimal. If the parameters are not well-initialized, the quality of the initial document retrievals would be low, leading to a feedback collection process that may not yield relevant documents. Consequently, the model might easily learn to distinguish these poorly retrieved documents from relevant ones, resulting in an inadequately trained search engine.

Furthermore, since the initial documents are retrieved without any prior feedback to adapt the search engine to each agent's information needs, they may not be relevant or useful to the agents. Additionally, relying solely on offline feedback collection limits the system, as it does not allow for continuous adaptation and improvement of the retrieval model based on real-time interactions. Implementing an online learning approach could address these issues by enabling the model to update its parameters incrementally during serving phase as new feedback is collected. This would allow the system to continuously refine its retrieval strategies, adjusting to the evolving information needs of each agent, improving personalization of the search engine for each agent.

We addresses the aforementioned issues by introducing the Iterative Utility Maximization (IUM) framework for training the retrieval model. This framework optimizes the retrieval model with the feedback collected from all agents during an offline phase, and it incorporates each agent's specific feedback during an online phase, all in an iterative manner to maximizes the probability of receiving positive feedback for each served query, as shown in Figure 1.

4.1 Offline Ranking Optimization through Iterative Utility Maximization

Similar to self-training of LLMs using self-generated data [49], we define a *optimality binary variable* o, where $p(o = 1|\mathcal{R}_k, x) \propto f_{M_i}(\mathcal{R}_k, x)$, meaning that variable $o = 1$ indicates positive feedback and $o = 0$ indicates negative feedback. Here, $\mathcal{R}_k$ is a retrieved list of k documents, and $f_{M_i}(\mathcal{R}_k, x)$ represents the feedback for the retrieved list $\mathcal{R}_k$ given input x from the RAG agent M_i. The main goal is to maximize log-likelihood of observing $o = 1$ for a given input x. Since in this process the retrieved documents affect the variable o, we can rewrite the log-likelihood as: $\log p(o = 1|x) = \log \sum_{\mathcal{R}_k \in \pi_k(C)} p_\theta(\mathcal{R}_k|x) p(o = 1|x, \mathcal{R}_k)$, where $\pi_k(C)$ denotes all permutations of k documents being selected from the corpus C. The

summation over all possible retrieval lists $\mathcal{R}_k$ is computationally expensive for a large corpus,[2] making the calculation infeasible. Instead of directly maximizing $\log p(o=1|x)$, one can maximize its Evidence Lower Bound (*ELBO*), denoted as $L(p_\theta, q)$, with respect to the retriever's parameters θ and a variational distribution $q(\mathcal{R}_k|x)$. The variational distribution is used to approximate the latent variable $\mathcal{R}_k$ distributions, simplifying and making the computation more efficient. Since $L(p_\theta, q)$ is a lower bound for $\log p(o=1|x)$, maximizing *ELBO* ensures increase in $\log p(o=1|x)$. Formally:

$$
\begin{aligned}
\log p(o=1|x) &= \log \sum_{\mathcal{R}_k \in \pi_k(C)} p_\theta(\mathcal{R}_k|x) p(o=1|x,\mathcal{R}_k) \times \frac{q(\mathcal{R}_k|x)}{q(\mathcal{R}_k|x)} \\
&= \log \mathop{\mathbb{E}}_{\mathcal{R}_k \sim q(\mathcal{R}_k|x)} \left[\frac{p(o=1|x,\mathcal{R}_k) p_\theta(\mathcal{R}_k|x)}{q(\mathcal{R}_k|x)} \right] \\
&\geq \mathop{\mathbb{E}}_{\mathcal{R}_k \sim q(\mathcal{R}_k|x)} \left[\log \frac{p(o=1|x,\mathcal{R}_k) p_\theta(\mathcal{R}_k|x)}{q(\mathcal{R}_k|x)} \right] \quad \text{(Jensen's ineq.)} \\
&= -D_{\mathrm{KL}}[q(\mathcal{R}_k|x) || p(o=1|x,\mathcal{R}_k) p_\theta(\mathcal{R}_k|x)] \\
&= \mathop{\mathbb{E}}_{\mathcal{R}_k \sim q(\mathcal{R}_k|x)} [\log p(o=1|x,\mathcal{R}_k)] - D_{\mathrm{KL}}[q(\mathcal{R}_k|x) || p_\theta(\mathcal{R}_k|x)] \\
&=: L(p_\theta, q) \qquad (1)
\end{aligned}
$$

where D_{KL} denotes the Kullback-Leibler divergence between the two given distributions. Thus, we show that $\log p(o=1|x) \geq L(p_\theta, q)$, which means that maximizing $L(p_\theta, q)$ results in increasing the lower bound of $\log p(o=1|x)$. To maximize $L(p_\theta, q)$, we utilize an Expectation-Maximization (EM) algorithm as follows.

Expectation Step ($q^{t+1} = \arg\max_q L(p_{\theta^t}, q)$)***:*** Maximizing q is considered the Expectation step since it involves finding the distribution q that best approximates the true posterior distribution of the latent variable $\mathcal{R}_k$. Considering the formulation in Equation 1, where $L(p_\theta, q) = -D_{\mathrm{KL}}[q(\mathcal{R}_k|x) || p(o=1|x,\mathcal{R}_k) p_{\theta^t}(\mathcal{R}_k|x)]$, it implies that maximum of $L(p_\theta, q)$ occurs when $q^{t+1} = p(o=1|x,\mathcal{R}_k) p_{\theta^t}(\mathcal{R}_k|x)$, because the KL divergence is non-negative and equals to zero when the given two distributions are identical.

Maximization Step ($\theta^{t+1} = \arg\max_\theta L(p_\theta, q^{t+1})$)***:*** The goal of this step is to update the model parameters by maximizing the expected log-likelihood computed in the Expectation step, thereby fit the model to the observed data. We re-write this step as:

$$
\begin{aligned}
\theta^{t+1} &= \arg\max_\theta L(p_\theta, q^{t+1}) = \arg\max_\theta -D_{\mathrm{KL}}[q^{t+1} || p_\theta(\mathcal{R}_k|x)] \\
&= \arg\max_\theta \sum_{\mathcal{R}_k} q^{t+1} \log p_\theta(\mathcal{R}_k|x) \\
&= \arg\max_\theta \sum_{\mathcal{R}_k} p(o=1|x,\mathcal{R}_k) p_{\theta^t}(\mathcal{R}_k|x) \log p_\theta(\mathcal{R}_k|x) \\
&= \arg\max_\theta \mathop{\mathbb{E}}_{\mathcal{R}_k \sim p_{\theta^t}(\mathcal{R}_k|x)} [p(o=1|x,\mathcal{R}_k) \log p_\theta(\mathcal{R}_k|x)] \qquad (2)
\end{aligned}
$$

Since feedback in our case is non-negative, as assumed in Section 3, and $p(o=1|x,\mathcal{R}_k) \propto f_{M_i}(\mathcal{R}_k, x)$, the final objective function considering all agents to get updated parameters θ^{t+1} is defined as:

$$
\arg\max_\theta \mathop{\mathbb{E}}_{M_i \sim M} \left[\mathop{\mathbb{E}}_{x \sim D_i^{\text{train}}} \left[\mathop{\mathbb{E}}_{\mathcal{R}_k \sim p_{\theta^t}(\mathcal{R}_k|x)} \left[f_{M_i}(\mathcal{R}_k, x) \log p_\theta(\mathcal{R}_k|x) \right] \right] \right] \qquad (3)
$$

[2]In our case, the corpus contains approximately 36 million document chunks.

where the retrieval lists are sampled from the parameters of the retrieval model in the previous iteration ($\mathcal{R}_k \sim p_{\theta^t}(\mathcal{R}_k|x)$). As noted by Salemi and Zamani [44], gathering agent's feedback for an entire ranked list is computationally expensive, and collecting feedback on a per-document basis is more efficient. Additionally, Singh et al. [50] emphasize the difficulty of calculating a retrieval list probabilities. To simplify this, we instead use a pointwise ranking objective function [28] using the same technique.

To implement this approach, two adjustments are needed: First, for pointwise learning-to-rank, sampling should be based on the probabilities of the documents, which can be approximated by their relevance to the query. Let $p_{\theta^t}(R=1|x,d)$ represent the probability that document d is relevant to query x. We assume that the more relevant a document is to a query, the higher its probability. Therefore, we have: $p^*_{\theta^t}(d|x) \propto p_{\theta^t}(R=1|x,d)$. Additionally, we assume that any documents outside the top k, ranked by $p_{\theta^t}(R=1|x,d)$, have a probability of zero. Thus, sampling from the retrieval list $\mathcal{R}k$ can be approximated by selecting the top k documents based on $p^*_{\theta^t}(d|x) \propto p_{\theta^t}(R=1|x,d)$. The second adjustment is that the objective function should aim to maximize the probability of documents that receive positive feedback. Thus, we can reformulate the problem by using the feedback as a relevance label, training the model to classify whether a document is relevant or not. We can define the final objective to get updated parameters θ^{t+1} as:

$$
\arg\max_\theta \mathop{\mathbb{E}}_{M_i \sim M} \left[\mathop{\mathbb{E}}_{x \sim D_i^{\text{train}}} \left[\mathop{\mathbb{E}}_{d \sim p^*_{\theta^t}(d|x)} \left[\log p_\theta(R = f_{M_i}(d,x)|x,d) \right] \right] \right] \qquad (4)
$$

Training Procedure: To train the retrieval model R_θ, we assume T expectation and maximization iterations are performed. In each iteration, the RAG agents submit their training queries to the retrieval model, which then provides them with the top results based on the given query from the corpus C. In response, the agents return feedback per retrieved document. After collecting feedback from all agents, the search engine enters an optimization phase where Equation 4 is used to optimize the model based on the provided feedback. This procedure is shown in Algorithm 1.

Algorithm 1 The Offline IUM algorithm.

Initialize the retrieval model from a pretrained encoder checkpoint
for $t = 1$ to T **do**
  $D^t = \{\}$
  for M_i in M **do**
    for x in D_i **do**
      Retrieve k docs from corpus C by R_{θ^t}: $r = R_{\theta^t}(x, C, k)$
      Collect feedback of agent M_i for list r: $f_i = f_{M_i}(d_i, x)$
      Add feedback to iteration t dataset: $D^t = D^t \cup \{(x, d_i, f_i)\}$
    end for
  end for
  $\theta^{t+1} = \arg\max_\theta \mathbb{E}_{(x,d,f)\sim D^t} [\log p_\theta(R = f|x,d)]$
end for

4.2 Online Ranking Optimization through Iterative Utility Maximization

In the previous subsection, R_θ collects feedback from all RAG agents on their respective training sets in an offline, iterative fashion, and uses their feedback for optimizing its parameters. However, this approach is not feasible during the serving phase[3] for several reasons. First, rapid access to information is necessary for the serving phase, since agents cannot afford to wait for feedback to be collected from all other agents. Additionally, offline optimization methods typically require access to the complete set of queries for effective optimization, which is not applicable in this context, where inputs appear sequentially. Therefore, this phase must rely on online optimization of the search engine tailored for each individual agent, utilizing their specific feedback. For this purpose, we assume that feedback for a test instance x_t from the agent M_i is only provided after R_θ has already retrieved and delivered the relevant documents to the agent, and the agent will not submit x_t to the system again afterwards. In other words, the search engine has only observed and served the first $t-1$ queries and received feedback for them by the time it attempts to serve x_t. Therefore, the search engine can utilize the feedback from the first $t-1$ queries to optimize its performance for serving the t^{th} query.

There are several strategies for updating the search engine parameters based on received online feedback. One possible approach is to update the search engine after each feedback to align it with the agent's preferences. However, this method has several shortcomings: first, updating the search engine after each feedback is computationally expensive. Additionally, the feedback for a single input may be noisy, leading to noisy gradient updates. On the other hand, an alternative strategy is to observe the majority of test instances before updating the model for the remaining queries. However, this would result in using the same old parameters for most of the test instances. Therefore, we adopt a middle-ground approach and apply the same method as described in Section 4.1 to optimize the model over several iterations. We define an iteration as serving b[4] consecutive queries and receiving the corresponding feedback. Thus, in step $t+1$, the model has access to the previous user queries $Q^t_{M_i} = \{x_1, ..., x_{b\times t}\}$ (This is the **Expectation Step**, as defined in Section 4.1). Finally, the parameters for agent M_i can be updated based on the following objective function (This is the **Maximization Step**, as defined in Section 4.1):

$$\theta^{t+1}_{M_i} = \arg\max_{\theta} \mathop{\mathbb{E}}_{x\sim Q^t_{M_i}} \left[\mathop{\mathbb{E}}_{d\sim p^*_{\theta^t_{M_i}}(d|x)} \left[\log p_\theta(R = f_{M_i}(d,x)|x,d) \right] \right] \tag{5}$$

where the main difference between this objective function and Equation 3 is that the optimization occurs for each individual agent based on their own feedback, with inputs sampled from the previous queries observed from the agent during the serving phase.

Training Procedure: To optimize the search engine using Online IUM, we assume a specific agent submits b queries to the search engine, which retrieves the top results from the corpus C for the given queries and the previous iteration parameters. In response, the agent provides feedback for the retrieved documents. Following this, the search engine enters an optimization phase, updating its parameters based on the feedback received for all queries submitted thus far by that agent, using Equation 5, to get the parameters for next iteration. This procedure is shown in Algorithm 2.

Algorithm 2 The Online IUM algorithm.

Initialize $t = 1$
Initialize the retrieval model for agent M_i from Algorithm 1 checkpoint
$D^{test}_{M_i} = \{\}$
while there is a query x_i **do**
  Retrieve k docs from corpus C by $R_{\theta^t}: r = R_{\theta^t_{M_i}}(x_i, C, k)$
  Collect feedback of agent M_i for list r: $f_j = f_{M_i}(d_j, x_i)$
  Add feedback to online dataset: $D^{test}_{M_i} = D^{test}_{M_i} \cup \{(x_i, d_j, f_j)\}$
  if $|D^{test}_{M_i}|$ dividable by b **then**
    $\theta^{t+1}_{M_i} = \arg\max_\theta \mathbb{E}_{(x,d,f)\sim D^{test}_{M_i}} [\log p_\theta(R = f|x,d)]$
    $t = t+1$
  end if
end while

4.3 The Search Engine Architecture

The proposed algorithms can work for any learning-to-rank approach. Following uRAG from Salemi and Zamani [45], this paper employs a two-stage cascaded retrieval system. In the first stage, BM25 [35] is used to retrieve a set of initial relevant candidate documents. Then, a fine-tuned cross-encoder model is applied to re-rank the retrieved documents. Note that all the trainable parameters θ in the search engine are for the re-ranking model. Following Nogueira and Cho [31], we add a linear projection over the representation of the start token (i.e., [CLS]) of a text encoder to obtain the relevance probability of a query and document, as follows:

$$p(R = 1|x,d) = \sigma(\text{ENCODER}(tid; mid; x; d) \cdot W) \tag{6}$$

where d is a document, x is the query, tid is an ID associated with the agent's task, mid is the ID associated with the LLM backbone used in the agent, σ is the sigmoid function, and $W \in \mathbb{R}^{D\times 1}$ is a linear projection, where D represents the embedding dimensionality of the encoder. We utilize BERT[5] [8] with 110M parameters and embedding dimensionality of $D = 768$ as the encoder. Here, following Salemi and Zamani [45], tid and mid is used for the purpose of personalizing retrieval model for each agent. In this way, each agent provides its underlying LLM architecture and the task it is performing, represented by a task ID (tid) and a model ID (mid). The search engine uses these identifiers as input and, when optimized with feedback from each user, learns both the specific preferences of individual users (presence of tid and mid together in input) and the general information needs associated with each task (tid) and architecture (mid). This enables the search engine to better understand user preferences and provide more effective, tailored results while also learning the general preferences associated with each task and agent architecture.

[3]The serving phase refers to the period when the system is being tested.
[4]We call this parameter the online optimization batch size, which differs from the batch size used in gradient optimization methods.
[5]Checkpoint can be find at: https://huggingface.co/google-bert/bert-base-uncased

Table 1: A list of RAG models used in our experiments for training and evaluation.

	Task	Data	Utility Func.	LM	#Docs
M_1	open-domain QA	NQ	Exact Match	RA-T5	10
M_2	open-domain QA	NQ	Exact Match	RA-BART	4
M_3	open-domain QA	NQ	Exact Match	FiD	10
M_4	open-domain QA	TriviaQA	Exact Match	RA-T5	10
M_5	open-domain QA	TriviaQA	Exact Match	RA-BART	4
M_6	open-domain QA	TriviaQA	Exact Match	FiD	10
M_7	open-domain QA	HotPotQA	Exact Match	RA-T5	10
M_8	open-domain QA	HotPotQA	Exact Match	RA-BART	4
M_9	open-domain QA	HotPotQA	Exact Match	FiD	10
M_{10}	fact verification	FEVER	Accuracy	RA-T5	10
M_{11}	fact verification	FEVER	Accuracy	RA-BART	4
M_{12}	fact verification	FEVER	Accuracy	FiD	10
M_{13}	slot filling	zsRE	Accuracy	RA-T5	10
M_{14}	slot filling	zsRE	Accuracy	RA-BART	4
M_{15}	slot filling	zsRE	Accuracy	FiD	10
M_{16}	slot filling	T-REx	Accuracy	RA-T5	10
M_{17}	slot filling	T-REx	Accuracy	RA-BART	4
M_{18}	slot filling	T-REx	Accuracy	FiD	10

Table 2: A list of datasets from KILT [32] used in our experiments. The validation data from KILT is used as test sets. * Given the large training set in the original T-REx dataset, we only sampled 5% of data for training our models.

Dataset	#train	#test
open-domain QA (short answer)		
Natural Questions	87,372	2,837
TriviaQA	61,844	5,359
HotPotQA	88,869	5,600
fact verification		
FEVER	104,966	10,444
slot-filling relation extraction		
zsRE	147,909	3,724
T-REx	114,208*	5,000

5 Experiments

5.1 Experiment Setup

Datasets & Corpus. We use various tasks from the KILT benchmark [32] to evaluate our approach. Dataset statistics is reported Table 2. We experiment with six diverse datasets, including three diverse open-domain question answering datasets: Natural Questions (NQ) [24], TriviaQA [19], and HotPotQA [60]. Notably, HotPotQA focuses on questions requiring multi-hop reasoning. Additionally, we use one fact verification dataset, FEVER [53], and two slot-filling datasets for relation extraction: zsRE [25] and T-REx [9]. Note that since the T-REx training set contains approximately 2.2 million samples, we randomly select 5% of them to train our models in order to speed up the experiments. We use the same samples as is used in [45]. It is also important to mention that the test set labels for these datasets are not publicly available. Therefore, we directly use the validation set to evaluate the models. Note that we utilize the McNemar statistical significance test [30] in our experiments, as the metrics for the tasks—accuracy and exact match—yield binary outcomes, making this test appropriate for evaluating significant performance differences. We use the Wikipedia dump provided by the KILT benchmark as the unstructured knowledge source.[6] We follow the pre-processing method outlined by Karpukhin et al. [20], in which each document is split into passages with a maximum length of 100 words. Additionally, the document title is concatenated with each passage to form the entries in the retrieval corpus.

Agents Configuration. Following Salemi and Zamani [45], we use 18 diverse RAG agents, as listed in Table 1. In this setting, each agent is trained on a separate dataset, with a distinct set of resources, a different underlying LLM, and retrieves a varying number of documents to perform its task. Each RAG agent is fine-tuned on the corresponding training set. Additionally, the number of retrieved documents is determined based on each agent's LLM maximum input size. We consider two types of RAG agents:

[6]The retrieval corpus is available at https://dl.fbaipublicfiles.com/ur/wikipedia_split/psgs_w100.tsv.gz

(1) Retrieval-augmented LLM (RA-X) is a language model that consumes k documents per input via in-prompt augmentation based on the following input format: "`{input} context 1: {doc1} ... context k: {dock}`", where `{input}` is x and `{doci}` denotes the content of the i^{th} retrieved document.
(2) Fusion-in-Decoder (FiD) [15] uses a different augmentation approach. Unlike RA-X that is based on in-prompt augmentation, FiD first encodes the input and each retrieved document separately and uses the concatenation of all document encodings as cross-attention for the decoder. FiD can thus only be done using encoder-decoder language models.

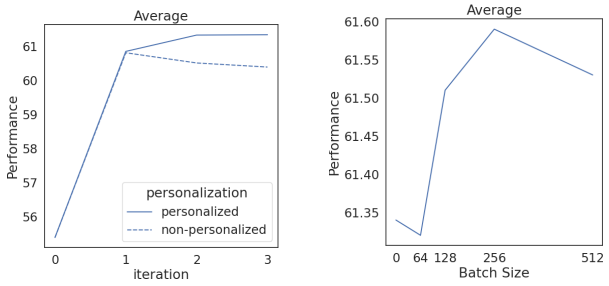
We utilize T5-small [33] with 60M parameters and BART-base [26] with 140M parameters using the first retrieval-augmentation approach and T5-small with the FiD. We use six datasets and apply three distinct RAG models to each, resulting in 18 unique agents for our experiments. All RAG agents are fine-tuned individually. The AdamW optimizer with a weight decay of 10^{-2} and a learning rate of 5×10^{-5} for 10 epochs is used to train these models. A linear warmup is applied to the first 5% of training steps. The effective batch size is set to 64 through gradient accumulation. Each model is trained on varying computational resources, including up to 8 A100, 1080ti, and 2080ti Nvidia GPUs. To optimize the RAG agents, in this paper, we employ a sequence-two-sequence loss function [52] using cross-entropy between the predicted token probability and the ground truth output sequence.

Each RAG agent is expected to provide feedback for the given retrieval list as a singal to help improving the search engine. For simplicity, in this paper, we consider the case where the RAG model provides feedback on a per-document basis. To generate feedback, we use the evaluation metric associated with the dataset that the model is applied to as the utility function for that model. Table 1 outlines the utility functions assigned to the RAG agents used in this study. Specifically, we employ Exact Match (EM) for question answering datasets and accuracy for the remaining tasks. For EM, we follow the post-processing procedures introduced by Rajpurkar et al. [34]. Accordingly, the feedback for a specific retrieval list reflects the performance of the RAG agent when using only each single document in the list in its task, and is evaluated based on the utility function and downstream task expected output.

Table 3: Downstream performance of RAG models in Table 1 utilizing the search engine. Superscripts 1, 2, 3, 4, and 5 denote statistically significant improvements in the performance compared to BM25, Contriever, $\text{Reranker}_{\text{individual}}$, $\text{Reranker}_{\text{dataset}}$, uRAG, respectively, using McNemar significance test ($p < 0.05$).

RAG Model	Data & Metric	BM25	Contriever	$\text{Reranker}_{\text{individual}}$	$\text{Reranker}_{\text{dataset}}$	uRAG	IUM	
							Offline IUM	Online IUM
M_1	NQ - EM	28.05	22.55	36.76	37.39	37.82	38.42^{12}	$\mathbf{38.84}^{1234}$
M_2	NQ - EM	33.09	23.68	40.07	40.50	42.12	43.60^{12345}	$\mathbf{45.11}^{12345}$
M_3	NQ - EM	29.64	23.69	40.50	41.14	42.37	$\mathbf{42.51}^{1234}$	42.47^{1234}
M_4	TriviaQA - EM	51.35	44.33	59.28	60.25	60.68	$\mathbf{61.87}^{12345}$	61.59^{12345}
M_5	TriviaQA - EM	57.52	48.49	64.76	67.23	68.12	68.03^{123}	$\mathbf{68.87}^{1234}$
M_6	TriviaQA - EM	60.48	49.30	67.44	68.63	68.74	$\mathbf{70.27}^{12345}$	70.05^{12345}
M_7	HotPotQA - EM	27.51	18.80	29.92	30.91	31.33	$\mathbf{31.41}^{123}$	31.16^{123}
M_8	HotPotQA - EM	31.21	20.78	35.03	34.62	34.85	$\mathbf{35.51}^{124}$	35.41^{12}
M_9	HotPotQA - EM	29.48	20.43	32.54	32.71	33.46	$\mathbf{33.77}^{1234}$	33.75^{1234}
M_{10}	FEVER - Accuracy	**86.83**	84.21	86.24	**86.83**	86.46	86.58^{2}	86.58^{2}
M_{11}	FEVER - Accuracy	**87.54**	84.37	84.38	**87.54**	85.99	86.17^{23}	87.44^{235}
M_{12}	FEVER - Accuracy	87.04	86.74	86.02	87.04	86.55	86.98^{235}	$\mathbf{87.47}^{235}$
M_{13}	zsRE - Accuracy	55.37	38.77	60.39	59.98	61.09	61.22^{1234}	$\mathbf{61.89}^{12345}$
M_{14}	zsRE - Accuracy	51.42	29.05	59.29	58.96	60.58	$\mathbf{60.68}^{1234}$	60.31^{1234}
M_{15}	zsRE - Accuracy	55.42	37.35	60.47	60.66	62.13	62.35^{1234}	$\mathbf{62.43}^{1234}$
M_{16}	T-REx - Accuracy	70.88	56.94	73.58	72.86	72.92	73.62^{12345}	$\mathbf{73.98}^{12345}$
M_{17}	T-REx - Accuracy	75.16	58.30	80.04	80.18	79.94	$\mathbf{80.32}^{12}$	80.24^{12}
M_{18}	T-REx - Accuracy	78.88	65.06	80.78	80.34	80.24	80.88^{125}	$\mathbf{81.14}^{12345}$
Overall	(macro-average)	55.38	45.15	59.86	60.43	60.85	61.34^{12345}	$\mathbf{61.59}^{12345}$

Search Engine Configuration. The search engine initially retrieves 100 documents using BM25 [35]. These documents are then reranked using the appropriate reranking model based on the experimental setup. Training R_θ involves two phase. For the first phase of training with Offline IUM, the Adam optimizer [22] with a learning rate of 10^{-5} for two epochs is used. A linear warmup is applied to the first 5% of training steps. The effective batch size is set to 512, achieved through gradient accumulation. A consistent value of $k = 32$ is used during training, and the maximum input length for this model is set to 256 tokens. We train the model for $T = 3$ iterations. Note that we randomly replace the model ID and task ID with the *"unk"* token for 10% of the training samples for better generalization. For the second phase training using Online IUM, we set batch size $b = 256$. Additionally, we utilize the same optimizer and learning rate to train the model for two epochs in each iteration. In this case, we set the variable k to be consistent with the downstream RAG agent configuration for each model in Table 1, in contrast with the offline phase, where we set a constant number of $k = 32$ for all agents. Additionally, we do not replace the model ID and task ID with the *"unk"* token for the sake of fully personalizing the search engine for the agent.



(a) The Effect of Offline Learning Number of Iterations **(b) The Effect of Online Learning Batch Size**

Figure 2: (a) The impact of iterations on personalized (solid) and non-personalized (dashed) Offline IUM and (b) the effect of batch size b on Online IUM performance.

5.2 Main Results

This section addresses the following research questions, providing detailed analyses and insights based on our experimental findings.

How does Offline IUM impact downstream RAG performance? We employ the baselines introduced by Salemi and Zamani [45] for comparative evaluation against our proposed approach. There are two types of baselines: (1) Single-stage retrieval and (2) Cascaded two-stage retrieval models. The single-stage retrieval baselines include BM25 [35] as a sparse retriever and Contriever [13] as a dense retriever. The cascaded two-stage methods employ BM25 in the first stage, followed by a reranker. Salemi and Zamani [45] introduces three baselines for reranking: 1) $\text{Reranker}_{\text{individual}}$, which is a reranker trained individually for each agent based on a single round of feedback from the respective agent, 2) $\text{Reranker}_{\text{dataset}}$ that is a reranker trained for each dataset similar to previous one, using feedback from all agents associated with that dataset, and 3) uRAG, which is a single reranker trained across all agents using the combined feedback from different agents. The results of this experiment are presented in Table 3 and suggest that Offline IUM outperforms all baselines for 14 out of 18 RAG agents, with the performance difference being statistically significant in 4 cases from all baselines. Moreover, Offline IUM achieves statistically significant improvements over all baselines when considering the average

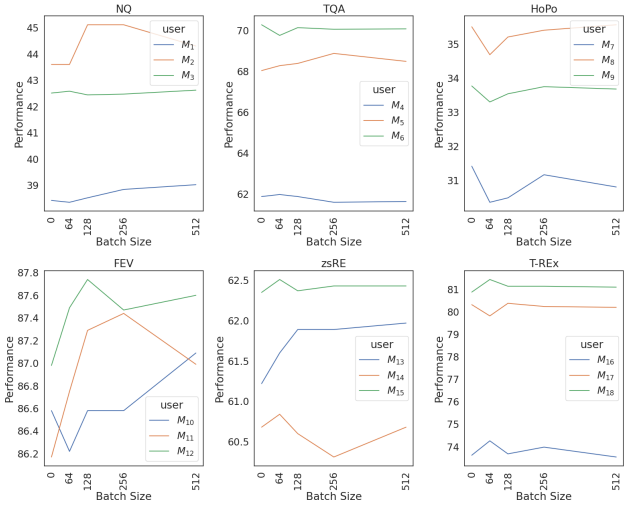


Figure 3: Effect of online learning batch size b on Online IUM on per agent performance.

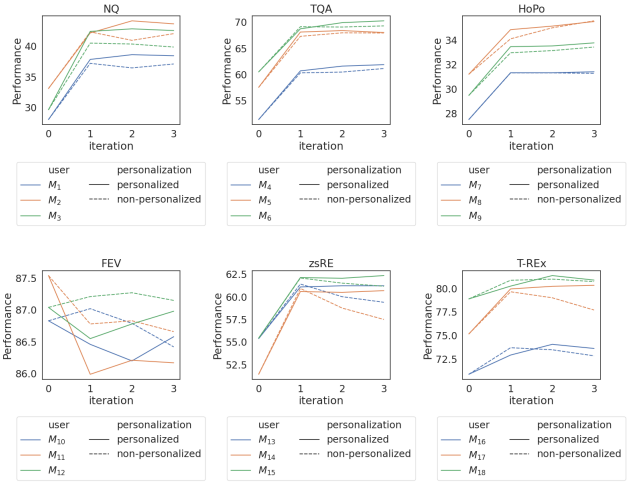


Figure 4: Impact of iteration count on personalized (solid) and non-personalized (dashed) Offline IUM plotted per agent.

performance across all agents. That said, the results from this experiment suggest that Offline IUM is a promising and effective approach for addressing the problem at hand.

How does the number of training iterations in Offline IUM impact the performance? An important hyperparameter for Offline IUM is the number of iterations. To investigate its impact, we train the search engine over several iterations and evaluate its performance at each step. The outcomes of this experiment are presented in Figure 2 (a—solid lines) for the average performance and Figure 4 for per agent performance. The results in Figure 2 (a) suggest that, overall, increasing the number of iterations leads to improved performance for Offline IUM. However, the magnitude of improvement diminishes with each additional iteration. For example, the improvement in average performance from iteration 0 to iteration 1 is 9%, while the gain from iteration 1 to 2 is 0.8%, and from iteration 2 to 3, the increase is less than 0.01%. This indicates diminishing returns as the number of iterations increases.

How does "personalization" in Offline IUM training phase impact the performance? As explained in Section 4, we use task ID (*tid*) and model ID (*mid*) to *personalize* the search engine for each agent during the Offline IUM training. In this experiment, we remove these IDs and train the search engine without them, using the same setup as Offline IUM. This allows us to examine the impact of these identifiers on personalization of the search engine during the Offline IUM training process on the overall performance. The results of this experiment are demonstrated in in Figure 2 (a) for average performance (dashed lines) and Figure 4 for per agent performance (dashed lines).

The results in Figure 4 indicate that training the search engine without incorporating task and model IDs leads to a decline in the performance with increasing iterations. Specifically, while most agents benefit from the initial training round, performance get worse for 10 out of 18 agents in the second iteration. Additionally, in Figure 2 (a), the average performance of the search engine also decreases with additional iterations. The results show that the performance gap between the personalized and non-personalized search engines widens with more iterations. Specifically, the plot reveals a growing divergence between the average performance of the personalized search engine and the non-personalized one as the number of iterations increases. This suggests that incorporating personalization significantly benefits the search engine, with the performance gains becoming more pronounced over time. We believe this phenomenon is likely due to the lack of personalization in the non-personalized search engine. Without task and model IDs, the search engine retrieves more generic documents in subsequent iterations, which may not be particularly useful for specific agents. Consequently, the search engine may begin to overfit to the average preferences of all agents during the offline phase, rather than effectively addressing the unique needs of each individual agent.

How does Online IUM impact downstream RAG performance? To investigate this question, we compare Online IUM with the previously discussed baselines. The results of this comparison are presented in Table 3. The results demonstrate that Online IUM surpasses the baselines for 13 out of 18 agents, with 6 of these differences being statistically significant. Moreover, Online IUM significantly enhances the average performance across all agents compared to the baseline methods. Another notable observation is that Online IUM enhances the performance of Offline IUM. Although this improvement is not statistically significant at the 95% confidence level, the p-value is quite small (p-value = 0.061), indicating a strong trend towards better performance. This observation suggests that applying Online IUM after Offline IUM is a promising method for the task at hand, indicating potential for further enhancing the system's performance.

How does the batch size b in Online IUM affects the performance? One of the key hyperparameters in Online IUM is the batch size b. This parameter determines how many consecutive queries the search engine should serve using the same set of parameters, essentially defining the feedback collection interval before

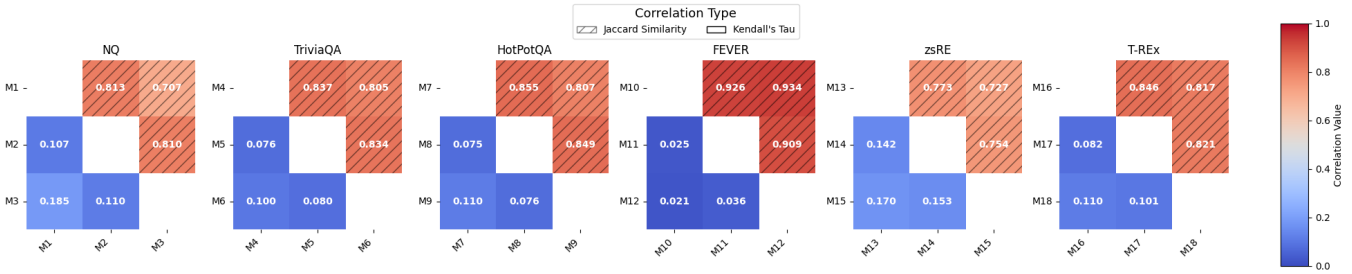


Figure 5: The Kendall's tau and Jaccard's similarity between the retrieval lists for agents that utilize the same dataset.

the model updates its parameters for the next batch. Exploring the impact of this parameter is insightful, as shown in Figure 2 (b) for average performance and Figure 3 for per agent performance. While the optimal batch size might vary depending on the agent, the general trend suggests that increasing the batch size improves performance up to a certain point, beyond which performance starts to decline. Notably, setting a small value (e.g., 64) leads to a performance drop compared to the starting checkpoint (i.e., the checkpoint from the Offline IUM approach). We believe this occurs because using a small batch size b leads to frequent updates in the model's parameters, making the model more susceptible to noisy feedback from the agent. These frequent updates might cause the model to overfit to specific feedback points rather than capturing generalizable patterns, thus negatively affecting performance.

Does "personalization" with IUM result in different retrieval lists for different RAG agents? We compare the retrieval lists for the same query across different agents when applying IUM to personalize the search engine for them. To evaluate the similarity between the retrieval lists, we employ two metrics: Kendall's tau coefficient and Jaccard's similarity. Kendall's tau coefficient captures how the ranking order of documents correlates between two lists, providing insights into how similarly or differently the documents are ranked for various agents. Jaccard's similarity, on the other hand, is a set-based metric that quantifies the overlap between two sets of retrieved documents, indicating the percentage of shared documents across retrieval lists for different agents. These metrics allow us to analyze both the ranking and the content of the retrieved documents across personalized search engine settings.

The results of this experiment are shown in Figure 5, yielding several key insights. First, the findings suggest that, on average, about 20% of the retrieved documents differ between RAG agents for the same query. This highlights the personalization effect, where each RAG agent receives a distinct set of documents despite querying with identical input. Furthermore, the low Kendall's tau correlation indicates significant differences in the ranking of the documents retrieved for different RAG agents, demonstrating that the search engine adapts document ranking based on agent-specific preferences and behaviors. Additionally, the Jaccard's similarity between agents employing FiD and those using in-prompt augmentation is notably lower than between agents both utilizing in-prompt augmentation. This demonstrates that the system has effectively learned to tailor document retrieval strategies according to the different retrieval-augmentation methods used by the RAG agents.

Another interesting observation is that the Kendall's tau correlation is higher between agents that both using T5 or FiD with T5 than between agents where one employs BART and the other utilizes T5 or FiD with T5. This suggests that the search engine, through model ID, has identified shared information needs between agents that utilize the same backbone language model (T5), resulting in more similar retrieval outputs. In summary, these results confirm that personalization significantly affects the retrieval lists provided to each agent, as the system learns to adapt document rankings and selections based on both the retrieval-augmentation method and the backbone LLM.

6 Conclusion & Future Work

In this paper, we address the challenge of building a search engine tailored for multiple RAG agents, functioning similarly to how search engines serve human users, considering the paradigm shift where nowadays LLMs are the main users of the search engines. We propose IUM, an iterative framework with expectation maximization to iteratively gather feedback from RAG agents and adjust the search engine based on them in an offline and online phase. Our findings demonstrate that the proposed approach statistically significantly outperforms established baselines in terms of average agents performance. We also conducted extensive studies to analyze the impact of key factors such as the number of training iterations in Offline IUM, batch size in Online IUM, and the role of personalization in search results for each agent. Overall, the proposed method exhibits promising results, showcasing its effectiveness in designing search engines for multiple RAG agents.

There are several potential future directions for this work: (1) extending the current setup to optimize retrieval models rather than just reranking; (2) considering multiple utility functions per agent; (3) investigating novel regularization techniques to enhance generalization for agents who do not participate in training; (4) exploring interleaving and counterfactual learning approaches within the context of a search engine for machines; and (5) expanding beyond text generation to address a more general REML scenario.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant number 2402873, and in part by the Office of Naval Research contract number N000142412612. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

[1] Eugene Agichtein, Eric Brill, and Susan Dumais. 2006. Improving web search ranking by incorporating user behavior information. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Seattle, Washington, USA) *(SIGIR '06)*. Association for Computing Machinery, New York, NY, USA, 19–26. https://doi.org/10.1145/1148170.1148177

[2] Garima Agrawal, Tharindu Kumarage, Zeyad Alghami, and Huan Liu. 2023. Can Knowledge Graphs Reduce Hallucinations in LLMs? : A Survey. arXiv:2311.07914 [cs.CL]

[3] 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*. https://openreview.net/forum?id=hSyW5go0v8

[4] Kevin Matthe Caramancion. 2024. Large Language Models vs. Search Engines: Evaluating User Preferences Across Varied Information Retrieval Scenarios. arXiv:2401.05761 [cs.IR] https://arxiv.org/abs/2401.05761

[5] Indu Chawla. 2011. An overview of personalization in web search. In *2011 3rd International Conference on Electronics Computer Technology*, Vol. 3. 128–132. https://doi.org/10.1109/ICECTECH.2011.5941815

[6] Wenhu Chen, Hexiang Hu, Xi Chen, Pat Verga, and William Cohen. 2022. MuRAG: Multimodal Retrieval-Augmented Generator for Open Question Answering over Images and Text. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 5558–5570. https://doi.org/10.18653/v1/2022.emnlp-main.375

[7] Ofer Dekel, Ran Gilad-Bachrach, Ohad Shamir, and Lin Xiao. 2012. Optimal distributed online prediction using mini-batches. *J. Mach. Learn. Res.* 13, null (jan 2012), 165–202.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Association for Computational Linguistics, Minneapolis, Minnesota, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[9] Hady Elsahar, Pavlos Vougiouklis, Arslen Remaci, Christophe Gravier, Jonathon Hare, Frederique Laforest, and Elena Simperl. 2018. T-REx: A Large Scale Alignment of Natural Language with Knowledge Base Triples. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Nicoletta Calzolari, Khalid Choukri, Christopher Cieri, Thierry Declerck, Sara Goggi, Koiti Hasida, Hitoshi Isahara, Bente Maegaard, Joseph Mariani, Hélène Mazo, Asuncion Moreno, Jan Odijk, Stelios Piperidis, and Takenobu Tokunaga (Eds.). European Language Resources Association (ELRA), Miyazaki, Japan. https://aclanthology.org/L18-1544

[10] Liangke Gui, Borui Wang, Qiuyuan Huang, Alexander Hauptmann, Yonatan Bisk, and Jianfeng Gao. 2022. KAT: A Knowledge Augmented Transformer for Vision-and-Language. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 956–968. https://doi.org/10.18653/v1/2022.naacl-main.70

[11] Katja Hofmann. 2013. Fast and reliable online learning to rank for information retrieval. *SIGIR Forum* 47, 2 (jan 2013), 140. https://doi.org/10.1145/2568388.2568413

[12] Steven C. H. Hoi, Doyen Sahoo, Jing Lu, and Peilin Zhao. 2018. Online Learning: A Comprehensive Survey. arXiv:1802.02871 [cs.LG] https://arxiv.org/abs/1802.02871

[13] Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised Dense Information Retrieval with Contrastive Learning. *Transactions on Machine Learning Research* (2022). https://openreview.net/forum?id=jKN1pXi7b0

[14] Gautier Izacard and Edouard Grave. 2020. Distilling Knowledge from Reader to Retriever for Question Answering. https://arxiv.org/abs/2012.04584

[15] Gautier Izacard and Edouard Grave. 2021. Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Association for Computational Linguistics, Online, 874–880. https://doi.org/10.18653/v1/2021.eacl-main.74

[16] Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. 2023. Atlas: Few-shot Learning with Retrieval Augmented Language Models. *Journal of Machine Learning Research* 24, 251 (2023), 1–43. http://jmlr.org/papers/v24/23-0037.html

[17] Thorsten Joachims and Filip Radlinski. 2007. Search Engines that Learn from Implicit Feedback. *Computer* 40, 8 (2007), 34–40. https://doi.org/10.1109/MC.2007.289

[18] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased Learning-to-Rank with Biased Feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining* (Cambridge, United Kingdom) *(WSDM '17)*. Association for Computing Machinery, New York, NY, USA, 781–789. https://doi.org/10.1145/3018661.3018699

[19] Mandar Joshi, Eunsol Choi, Daniel 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)*, Regina Barzilay and Min-Yen Kan (Eds.). Association for Computational Linguistics, Vancouver, Canada, 1601–1611. https://doi.org/10.18653/v1/P17-1147

[20] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Online, 6769–6781. https://doi.org/10.18653/v1/2020.emnlp-main.550

[21] To Eun Kim, Alireza Salemi, Andrew Drozdov, Fernando Diaz, and Hamed Zamani. 2024. Retrieval-Enhanced Machine Learning: Synthesis and Opportunities. arXiv:2407.12982 [cs.LG] https://arxiv.org/abs/2407.12982

[22] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG]

[23] Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, and Hamed Zamani. 2024. LongLaMP: A Benchmark for Personalized Long-form Text Generation. arXiv:2407.11016 [cs.CL] https://arxiv.org/abs/2407.11016

[24] 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 (2019), 452–466. https://doi.org/10.1162/tacl_a_00276

[25] Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-Shot Relation Extraction via Reading Comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, Roger Levy and Lucia Specia (Eds.). Association for Computational Linguistics, Vancouver, Canada, 333–342. https://doi.org/10.18653/v1/K17-1034

[26] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 7871–7880. https://doi.org/10.18653/v1/2020.acl-main.703

[27] 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 *Proceedings of the 34th International Conference on Neural Information Processing Systems* (Vancouver, BC, Canada) *(NIPS'20)*. Curran Associates Inc., Red Hook, NY, USA, Article 793, 16 pages.

[28] Tie-Yan Liu. 2009. Learning to Rank for Information Retrieval. *Found. Trends Inf. Retr.* 3, 3 (March 2009), 225–331. https://doi.org/10.1561/1500000016

[29] Bryan McCann, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher. 2019. The Natural Language Decathlon: Multitask Learning as Question Answering. https://openreview.net/forum?id=B1lfHhR9tm

[30] Quinn McNemar. 1947. Note on the sampling error of the difference between correlated proportions or percentages. *Psychometrika* 12, 2 (1947), 153–157. https://doi.org/10.1007/BF02295996

[31] Rodrigo Nogueira and Kyunghyun Cho. 2020. Passage Re-ranking with BERT. arXiv:1901.04085 [cs.IR]

[32] Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a Benchmark for Knowledge Intensive Language Tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 2523–2544. https://doi.org/10.18653/v1/2021.naacl-main.200

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (jan 2020), 67 pages.

[34] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Austin, Texas, 2383–2392. https://doi.org/10.18653/v1/D16-1264

[35] Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Text Retrieval Conference*. https://api.semanticscholar.org/CorpusID:3946054

[36] Devendra Sachan, Mostofa Patwary, Mohammad Shoeybi, Neel Kant, Wei Ping, William L. Hamilton, and Bryan Catanzaro. 2021. End-to-End Training of Neural

Retrievers for Open-Domain Question Answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Online, 6648–6662. https://doi.org/10.18653/v1/2021.acl-long.519

[37] Alireza Salemi, Juan Altmayer Pizzorno, and Hamed Zamani. 2023. A Symmetric Dual Encoding Dense Retrieval Framework for Knowledge-Intensive Visual Question Answering. In *Proceedings of the 46th International ACM SIGIR Conference on Research and Development in Information Retrieval* (<conf-loc>, <city>Taipei</city>, <country>Taiwan</country>, </conf-loc>) *(SIGIR '23)*. Association for Computing Machinery, New York, NY, USA, 110–120. https://doi.org/10.1145/3539618.3591629

[38] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024. Optimization Methods for Personalizing Large Language Models through Retrieval Augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 752–762. https://doi.org/10.1145/3626772.3657783

[39] Alireza Salemi, Julian Killingback, and Hamed Zamani. 2025. ExPerT: Effective and Explainable Evaluation of Personalized Long-Form Text Generation. arXiv:2501.14956 [cs.CL] https://arxiv.org/abs/2501.14956

[40] Alireza Salemi, Cheng Li, Mingyang Zhang, Qiaozhu Mei, Weize Kong, Tao Chen, Zhuowan Li, Michael Bendersky, and Hamed Zamani. 2025. Reasoning-Enhanced Self-Training for Long-Form Personalized Text Generation. arXiv:2501.04167 [cs.CL] https://arxiv.org/abs/2501.04167

[41] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When Large Language Models Meet Personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Lun-Wei Ku, Andre Martins, and Vivek Srikumar (Eds.). Association for Computational Linguistics, Bangkok, Thailand, 7370–7392. https://aclanthology.org/2024.acl-long.399

[42] Alireza Salemi, Mahta Rafiee, and Hamed Zamani. 2023. Pre-Training Multi-Modal Dense Retrievers for Outside-Knowledge Visual Question Answering. In *Proceedings of the 2023 ACM SIGIR International Conference on Theory of Information Retrieval* (Taipei, Taiwan) *(ICTIR '23)*. Association for Computing Machinery, New York, NY, USA, 169–176. https://doi.org/10.1145/3578337.3605137

[43] Alireza Salemi and Hamed Zamani. 2024. Comparing Retrieval-Augmentation and Parameter-Efficient Fine-Tuning for Privacy-Preserving Personalization of Large Language Models. arXiv:2409.09510 [cs.CL] https://arxiv.org/abs/2409.09510

[44] Alireza Salemi and Hamed Zamani. 2024. Evaluating Retrieval Quality in Retrieval-Augmented Generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 2395–2400. https://doi.org/10.1145/3626772.3657957

[45] Alireza Salemi and Hamed Zamani. 2024. Towards a Search Engine for Machines: Unified Ranking for Multiple Retrieval-Augmented Large Language Models. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 741–751. https://doi.org/10.1145/3626772.3657733

[46] Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. ColBERTv2: Effective and Efficient Retrieval via Lightweight Late Interaction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 3715–3734. https://doi.org/10.18653/v1/2022.naacl-main.272

[47] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval Augmentation Reduces Hallucination in Conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*. Association for Computational Linguistics, Punta Cana, Dominican Republic, 3784–3803. https://doi.org/10.18653/v1/2021.findings-emnlp.320

[48] Ahu Sieg, Bamshad Mobasher, and Robin Burke. 2007. Web search personalization with ontological user profiles. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management* (Lisbon, Portugal) *(CIKM '07)*. Association for Computing Machinery, New York, NY, USA, 525–534. https://doi.org/10.1145/1321440.1321515

[49] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond Human Data: Scaling Self-Training for Problem-Solving with Language Models. *Transactions on Machine Learning Research* (2024). https://openreview.net/forum?id=lNAyUngGFK Expert Certification.

[50] Devendra Singh, Siva Reddy, Will Hamilton, Chris Dyer, and Dani Yogatama. 2021. End-to-End Training of Multi-Document Reader and Retriever for Open-Domain Question Answering. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.), Vol. 34. Curran Associates, Inc., 25968–25981. https://proceedings.neurips.cc/paper_files/paper/2021/file/da3fde159d754a2555eaa198d2d105b2-Paper.pdf

[51] Shamane Siriwardhana, Rivindu Weerasekera, Elliott Wen, Tharindu Kaluarachchi, Rajib Rana, and Suranga Nanayakkara. 2023. Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering. *Transactions of the Association for Computational Linguistics* 11 (2023), 1–17. https://doi.org/10.1162/tacl_a_00530

[52] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2* (Montreal, Canada) *(NIPS'14)*. MIT Press, Cambridge, MA, USA, 3104–3112.

[53] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a Large-scale Dataset for Fact Extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, New Orleans, Louisiana, 809–819. https://doi.org/10.18653/v1/N18-1074

[54] Nuno Vasconcelos and Andrew Lippman. 1999. Learning from User Feedback in Image Retrieval Systems. In *Advances in Neural Information Processing Systems*, S. Solla, T. Leen, and K. Müller (Eds.), Vol. 12. MIT Press. https://proceedings.neurips.cc/paper_files/paper/1999/file/7283518d47a05a09d33779a17adf1707-Paper.pdf

[55] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2019. *SuperGLUE: a stickier benchmark for general-purpose language understanding systems*. Curran Associates Inc., Red Hook, NY, USA.

[56] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Association for Computational Linguistics, Brussels, Belgium, 353–355. https://doi.org/10.18653/v1/W18-5446

[57] Dingmin Wang, Qiuyuan Huang, Matthew Jackson, and Jianfeng Gao. 2024. Retrieve What You Need: A Mutual Learning Framework for Open-domain Question Answering. *Transactions of the Association for Computational Linguistics* 12 (2024), 247–263. https://doi.org/10.1162/tacl_a_00646

[58] Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2024. Text Embeddings by Weakly-Supervised Contrastive Pre-training. arXiv:2212.03533 [cs.CL] https://arxiv.org/abs/2212.03533

[59] Sohee Yang and Minjoon Seo. 2020. Is Retriever Merely an Approximator of Reader? arXiv:2010.10999 [cs.CL]

[60] 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*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (Eds.). Association for Computational Linguistics, Brussels, Belgium, 2369–2380. https://doi.org/10.18653/v1/D18-1259

[61] Hamed Zamani and Michael Bendersky. 2024. Stochastic RAG: End-to-End Retrieval-Augmented Generation through Expected Utility Maximization. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 2641–2646. https://doi.org/10.1145/3626772.3657923

[62] Hamed Zamani, Fernando Diaz, Mostafa Dehghani, Donald Metzler, and Michael Bendersky. 2022. Retrieval-Enhanced Machine Learning. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (, Madrid, Spain,) *(SIGIR '22)*. Association for Computing Machinery, New York, NY, USA, 2875–2886. https://doi.org/10.1145/3477495.3531722