arXiv:2405.00175v1 [cs.CL] 30 Apr 2024

Towards a Search Engine for Machines: Unified Ranking for Multiple Retrieval-Augmented Large Language Models

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 introduces uRAG–a framework with a unified retrieval engine that serves multiple downstream retrieval-augmented generation (RAG) systems. Each RAG system consumes the retrieval results for a unique purpose, such as open-domain question answering, fact verification, entity linking, and relation extraction. We introduce a generic training guideline that standardizes the communication between the search engine and the downstream RAG systems that engage in optimizing the retrieval model. This lays the groundwork for us to build a large-scale experimentation ecosystem consisting of 18 RAG systems that engage in training and 18 unknown RAG systems that use the uRAG as the new users of the search engine. Using this experimentation ecosystem, we answer a number of fundamental research questions that improve our understanding of promises and challenges in developing search engines for machines.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; **Machine learning**; • **Information systems** → **Information retrieval**; **Retrieval models and ranking**; **Question answering**.

KEYWORDS

Retrieval-enhanced machine learning; retrieval augmentation; neural ranking model; large language model; text generation

ACM Reference Format:
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 (SIGIR '24), July 14–18, 2024, Washington, DC, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3626772.3657733

1 INTRODUCTION

The vast majority of machine learning systems, including large generative models, are designed as self-contained systems, with both knowledge and reasoning encoded in model parameters. However, these models cannot work effectively for tasks that require knowledge grounding [1, 48], especially in case of non-stationary data where new information is actively being produced [49, 57]. As suggested by Zamani et al. [57], this issue can be addressed when machine learning systems are being *enhanced with the capability of retrieving stored content*. For example, in retrieval-augmented generation (RAG), as a special case of retrieval-enhanced machine learning (REML) [57], systems consume the responses provided by a retrieval model for the purpose of text generation [25, 26]. RAG models demonstrate substantial promise across various applications, including open-domain question answering [6, 20, 25, 44, 60], fact verification [46], dialogue systems [8, 45, 53], machine translation [4], and personalized generation [37, 38].

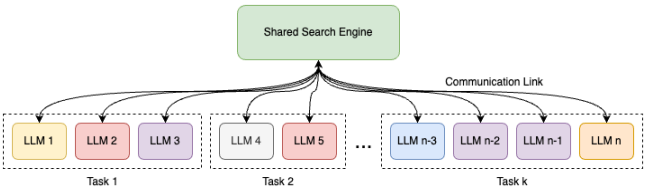

Figure 1: A high-level overview of the uRAG ecosystem. The ecosystem consists of a shared search engine that serves multiple RAG models, each performing its own task.

In the RAG literature, the retrieval component is often implemented using either of the following two approaches:

(1) Employing an off-the-shelf retrieval model that does not require training for the downstream RAG system: in this category, RAG systems either use APIs from commercial web search engines [29], term matching retrieval models [11], such as TF-IDF and BM25, or neural ranking models trained on relevance annotations provided as an external resource [25];
(2) Training a retrieval model given the feedback from the downstream RAG system through knowledge distillation [15] or end-to-end optimization [35].

As expected, the latter category offers the current state-of-the-art performance for various tasks [17, 52]. From an IR perspective, in this category, the downstream RAG model is the only "user" of the search engine. This paper takes a step further by relying on an important lesson learned by the IR community since the creation of web search: *achieving an effective retrieval model can be achieved through aggregating large-scale implicit feedback (e.g., clicks) obtained across users* [42]. Following this lesson, we aim to develop uRAG, a framework for training a single unified search engine for multiple RAG systems conducting various downstream tasks (see Figure 1). We define a training guideline, in which the search engine and its users—multiple downstream RAG systems—engage in an optimization process, in which RAG systems submit queries and the

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
SIGIR '24, July 14–18, 2024, Washington, DC, USA
© 2024 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0431-4/24/07.
https://doi.org/10.1145/3626772.3657733

search engine solicits feedback for a number of returned documents in their response. Each downstream RAG model identifies a utility function (any arbitrary evaluation metric; differentiable or not) and uses it for providing scalar feedback to the search engine.

Using such a generic optimization guideline by uRAG, we implement a large-scale experimentation ecosystem that enables addressing important research questions in this area. In more detail, we implement a set of 18 RAG systems that use different large language model (LLM) architectures, conduct different tasks, consume different number of retrieved documents, and/or trained on different datasets. We consider three diverse tasks of open-domain question answering, fact verification, and slot-filling for relation extraction. Six datasets related to these tasks from the knowledge-intensive language tasks (KILT) benchmark [31] are used for training and evaluation. In addition to these 18 RAG models that engage with our unified search engine for optimization, we consider another set of 18 RAG models with distinct properties as the *new users* of uRAG that do not engage in optimization and are partially or entirely unknown to the search engine. These models may use different LLM architectures, introduce new tasks, and/or use a new dataset for a known task. Using such a rich ecosystem, we aim at answering the following fundamental research questions that we believe are essential to understand the potential of developing *search engines for machines*. Note that given the scope of these research questions and space limitation, we limit this study to reranking optimization, in which the search engine retrieves documents using BM25 and optimizes a personalized cross-encoder reranker based on the feedback received from the downstream RAG models.

RQ1: How does unified reranking perform compared to training individual rerankers for each RAG model? Our experiments demonstrated that unified reranking performs either on par or significantly better than the same reranking model being trained for each downstream RAG model individually. In more detail, improvements are statistically significant for 61% of downstream models and there is no statistically significant performance degradation across the 18 RAG models that engage in the training process. This critical finding suggests that developing a unified search engine for machines is a promising research direction for the IR community to pursue further.

RQ2: How does unified reranking perform compared to training rerankers using the feedback obtained from all RAG systems that use the same dataset? This research question helps us understand if unification across datasets and tasks can lead to downstream performance improvement. Our experiments suggest that again training a unified reranker performs on par or significantly better than the ones that are trained based on the feedback provided by all RAG models that use the same dataset. In more detail, 39% of downstream RAG models in our experiments observe statistically significant improvement, while the majority observe no significant differences. These results are particularly important as they show knowledge transfer can occur across datasets.

RQ3: How does "personalizing" search results for different RAG systems impact the performance? We hypothesize that different RAG systems behave differently, thus "personalizing" search results for each RAG model can be beneficial. To achieve this, we feed a task and model identifier for each RAG model to our cross-encoder reranker, in addition to the query and document content. We observe that 50% of downstream RAG models observe improvements by including task and model identifiers in relevance scoring. However, in the vast majority of cases, these improvements are not statistically significant. Therefore, we do not find personalization in our experimentation ecosystem substantially useful. We must acknowledge that personalization often demonstrates its significance as the number of users increases and here we only have 18 users (i.e., the downstream RAG systems that engage in training). Therefore, this leaves several open questions for future work on exploring personalization in the context of uRAG.

RQ4: How does unified reranking perform for new (unknown) RAG systems that use an observed (known) dataset? Ideally, search engines should perform effectively and robustly for new users. This would demonstrate the generalizability of the retrieval model to unknown agents. To address this question, we introduce three retrieval-augmented generation models based on PEGASUS [58], Mistral [18], and Llama2 [47] to our experiments. These models did not engage in the optimization pipeline of uRAG, thus are unknown to our unified reranker. We demonstrate that unified reranking significantly outperforms BM25 for all these new (unknown) RAG systems and performs comparably with a reranker that are trained on all other RAG systems that use the same dataset.

RQ5: How does unified reranking perform for a known RAG system on a new (unknown) dataset that is similar to the ones used during training? To address this research question, we use the open-domain variant of the SQuAD dataset [33] as a question answering data with short answers that is relatively similar to some of the datasets used during training. We observe that unified reranking leads to significant improvements compared to BM25 and since the dataset is new, other reranking alternatives are not applicable to this scenario.

RQ6: How does unified reranking perform for a known RAG system on entirely new tasks? We take a step further and consider evaluation sets that are not closely related to our training datasets. For this purpose, we consider ELI5, an open-domain question answering dataset with long-form (passage-level) answers, as well as AY2 an entity linking dataset. Note that all datasets used during optimization target short text generation tasks. We observe that uRAG outperforms BM25 in half of the cases and differences are not statistically significant. This suggests that reranking documents using the current implementation of uRAG should only be employed when target downstream tasks are closely related to the ones used during training.

RQ7: How does unified reranking perform for a new (unknown) RAG system on a new (unknown) dataset that is similar to the ones used during training? We evaluate retrieval-augmented models based on PEGASUS, Mistral, and Llama2 on SQuAD for answering this question. None of these models were employed during training and as mentioned earlier, SQuAD is similar to some of the datasets used for training uRAG. We observe that reranking results using uRAG can lead to statistically significant improvements in this scenario. This emphasizes that the target task

and data have more impact on the effectiveness of uRAG, compared to the downstream RAG system that consumes the retrieval results.

RQ8: How does unified reranking perform for a new RAG system on an entirely new (unknown) task? This is the most extreme case, where both the model architecture and the task are unknown to the search engine. We test PEGASUS, Mistral, and Llama2 on ELI5 and AY2 and observe that uRAG struggles with improving BM25 in this scenario. We mark the development of robust and generalizable search engines for such scenarios as an important research direction for further exploration.

RQ9: How does uRAG perform when different amount of training data is provided? We plot the learning curve for all 18 RAG models that are involved in our training pipeline and observe that their performance often improves as more training data is provided. This finding suggests that as we introduce more data and more RAG systems to the uRAG framework, we expect even further performance improvement.

We believe that these findings deepen our knowledge and understanding for developing effective search engines for downstream retrieval-augmented systems and smooth the path towards important developments in this area. We open-source the uRAG codebase and release our trained model parameters.[1] Given the high cost of training and building such an ecosystem, we expect that this public release will substantially speed up research progress in this area.

2 RELATED WORK

Knowledge-Intensive Language Tasks. Unlike conventional NLP tasks like natural language understanding [50, 51] and question answering [28], where the task's input alone is sufficient for task completion, knowledge-intensive NLP tasks demand access to external knowledge sources to retrieve essential information for task execution. Petroni et al. [31] established KILT as a benchmark for knowledge-intensive NLP tasks. Encompassing a wide range of tasks such as open-domain question answering, fact checking, slot filling, and entity linking, KILT provides a comprehensive evaluation framework. The experiments conducted in this paper leverage datasets sourced from the KILT benchmark.

Retrieval-Augmented Generation (RAG). RAG [25] is a paradigm that integrates information retrieval (IR) and natural language generation (NLG). This approach aims to enhance the quality and relevance of generated content by leveraging external knowledge sources during the generation process [3, 44]. Unlike traditional large language models that rely solely on knowledge learned during their pre-training, RAG systems dynamically fetch information from a knowledge source using a retriever, allowing them to generate contextually rich and factually accurate outputs [57]. RAG demonstrates versatility with numerous use cases and applications. It finds application in knowledge-grounding within textual [16, 25, 31] or multi-modal [5, 12, 36, 39] problem domains. Additionally, RAG contributes to personalization efforts [37, 38] and addresses challenges such as reducing hallucination [2, 43].

[1]https://github.com/alirezasalemi7/uRAG

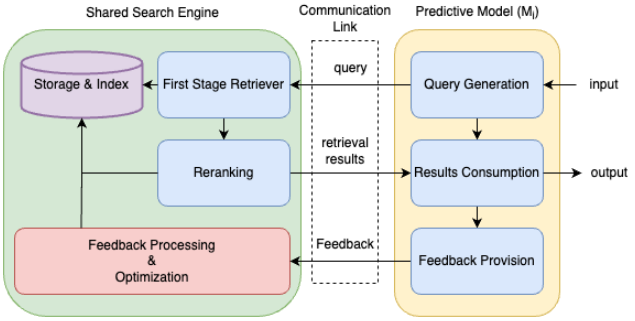

Figure 2: An overview of interactions between RAG models (also known as predictive models) and the unified search engine.

A pivotal element in RAG systems is the retriever, tasked with obtaining the required information for the large language model to execute its task [25]. Typically, a either sparse (e.g., BM25 [34]) or dense retrieval models (e.g., DPR [20], Contriever [14], ColBERTv2 [41], etc.) is employed to efficiently retrieve information from knowledge sources. Subsequently, the large language model employs the retrieved information to accomplish its task. Notably, In-Prompt Augmentation (IPA) and Fusion-in-Decoder (FiD) [16] represent prominent approaches in this context. In IPA, the retrieved information is concatenated with the prompt, enabling the large language model to fulfill the task. On the other hand, FiD involves encoding each retrieved document and the prompt separately in the encoder of an encoder-decoder LLM, followed by concatenation in the decoder to generate a unified answer drawing from the information in all documents. For a more in-depth exploration of this approach, refer to the work by Izacard and Grave [16].

Optimizing Retriever in RAG pipelines. Joint training of a retriever model and a large language model has been explored in various studies. Yang and Seo [54] focuses on optimizing the retriever by leveraging the downstream performance of the LLM, using each individually retrieved document to guide the retriever towards retrieving documents with higher scores. Conversely, Izacard and Grave [15] adopts a different strategy by utilizing cross-attention weights of the LLM instead of the downstream performance to distill knowledge from the LLM to the retriever. Additionally, Sachan et al. [35] introduces EMDR^2, an end-to-end approach that employs an expectation-maximization algorithm to optimize both the retriever and the LLM, maximizing the probability of generating the ground truth answer conditioned on the documents retrieved by the retriever. Moreover, Izacard et al. [17] presents a similar approach to Izacard and Grave [15] and Sachan et al. [35], utilizing the document's posterior distribution according to the LLM, conditioned on a single document, to perform distillation. Wang et al. [52] uses a multi-armed bandit algorithm to for the joint training of a retriever and reader. Most recently, Zamani and Bendersky [56] proposes end-to-end RAG optimization through stochastic sampling without replacement, approximated by gumbel-top-k sampling.

Our work diverges from prior research in two key aspects. Firstly, we operate under the assumption that LLM parameters in downstream models are not accessible to the retrieval model. This stems

from our conceptualization of the LLM as a user of the retrieval model, akin to how humans interact with search engines. In this analogy, search engines can only offer results to users and must rely on user feedback (e.g., clicks) to refine themselves, without altering how users engage with the results. Secondly, our focus extends beyond previous studies by considering scenarios where multiple LLMs use a shared search engine. Our objective is to optimize the search engine to enhance the collective performance of these diverse LLMs.

3 THE URAG ECOSYSTEM

Search engines are often designed for human users who submit unstructured queries, such as keyword queries and natural language questions. However, a paradigm shift is underway with the emergence of machine learning models [57], especially large language models, possessing strong linguistic capabilities, memorization, and sometimes even reasoning to some extent [59]. In the current paradigm known as REML [57], machines, e.g., LLMs, engage in interactions with a retrieval model to acquire essential knowledge for executing their designated tasks. Most REML systems nowadays are in the form of retrieval-augmented text generation. Notably, with the introduction of real-world RAG systems, such as Bing Chat[2] and Google Bard,[3] humans now assume the role of users for these applications [40, 57], marking a transition from their previous role as users of web search engines. In this case, LLMs constitute the primary users of search engines. Exploring the design of a search engine capable of providing information to different LLMs based on their needs, each tailored for specific tasks, represents a valuable and promising research direction that this paper focuses on.

uRAG Formulation. Let R_θ denote a search engine parameterized by θ whose goal is to provide access to information from a corpus C to a set of n RAG models. The set of RAG models, that are essentially the users of R_θ, is represented as $\mathbf{M} = \{M_1, ..., M_n\}$. For R_θ, each M_i is a black-box model and the search engine does not have access to the architecture, configuration, or parameters of RAG models. Each RAG model M_i is designed to perform a specific task T_i that requires access to information from the corpus C. These tasks are often called knowledge-intensive tasks. Each RAG model M_i operates by taking the input x and utilizing a communication link to interact with the search engine R_θ. Through this interaction, the model produces an output denoted as $\hat{y}_{M_i}$, representing the result of the prediction process for the given input x. Formally, $\hat{y}_{M_i} = M_i(x; R_\theta)$. Inspired by the REML framework presented in [57], Figure 2 illustrates the interactions and components in uRAG.

The Optimization Guideline and Process in uRAG. For each query submitted by the RAG model M_i, our goal is to develop a search engine that delivers a search result that benefits the downstream task conducted by M_i. Thus, we assume that M_is engage in an optimization process with R_θ with a pre-defined training guideline:

(1) Each RAG model M_i must have a training set $\mathbf{D_i} = \{(x_1, y_1), \cdots, (x_{|\mathbf{D_i}|}, y_{|\mathbf{D_i}|})\}$, where (x_j, y_j) denotes the jth input and expected output. There is no need to share the data with the search engine R_θ.
(2) For each $(x, y) \in \mathbf{D}_i$, the RAG model M_i must construct a query $q = \text{QGen}(x)$ to be consumed by the search engine R_θ. The search engine returns a ranked list of documents in response. Let $\mathbf{Q}_i = \{\text{QGen}(x) : \forall (x, y) \in \mathbf{D}_i\}$ be a set of all queries constructed from data points in $\mathbf{D}_i$.
(3) Each RAG model M_i must identify a utility function Utility_i that quantifies the end-to-end performance of the model in relation to its designated task T_i and serves as a metric to measure the effectiveness of M_i in performing its downstream task. Utility_i can be any arbitrary function, differentiable or not, whose outputs will be provided as feedback to R_θ for optimization. For simplicity, this paper assumes that Utility_i values are in the $[0, 1]$ range.

Table 1: A list of datasets from KILT [31] used in our experiments. The validation data from KILT is used as test sets. The first six datasets are used during the search engine training process. The last three datasets, on the other hand, are only introduced during inference to quantify generalizability of uRAG. * 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
new data introduced during inference		
SQuAD	87,599	10,570
ELI5	272,634	1,507
AY2	18,395	4,784

According to the risk minimization framework, the primary goal of the search engine R_θ is to minimize the loss function defined based on the received utility values from downstream models:

$$\theta^* = \arg\min_\theta \frac{1}{n} \sum_{i=1}^{n} \frac{1}{|\mathbf{Q}_i|} \sum_{q \in \mathbf{Q}_i} L(R_\theta(q; C), U_{iq}) \tag{1}$$

where U_{iq} is computed by M_i as follows:

$$U_{iq} \coloneqq \text{Utility}_i(y, M_i(x; R_\theta(\text{QGen}_i(x); C))) \tag{2}$$

It is important to note that various methods exist for aggregating utility across the users. For example, prioritizing utility enhancement for specific users may be more important than others. Alternatively, fairness or robustness could serve as criteria for aggregating the overall system utility. However, in this particular problem, our emphasis is on expected utility across inputs and users.

[2]https://www.bing.com/chat
[3]https://bard.google.com/

Table 2: A list of RAG models used in our experiments for training and evaluation.

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

4 SYSTEM IMPLEMENTATION AND SETUP

4.1 Training Data

As we will discuss later, different tasks and datasets are used for training RAG models and providing feedback to R_θ. A list of all datasets is presented in Table 1. Three diverse open-domain question answering datasets, i.e., Natural Questions (NQ) [22], TriviaQA [19], and HotPotQA [55], are used in our experiments. The answers for all questions in these datasets are in the form of short text. HotPotQA focuses on questions that require multi-hop reasoning. One dataset is used for fact verification, called FEVER [46], and finally, two slot-filling datasets for relation extraction are used in our experiments; zsRE [23] and T-REx [9]. All datasets are obtained from the KILT benchmark [31]. Note that the last three datasets mentioned in Table 1 are not used for optimizing R_θ and they are primarily employed for evaluating the generalizability of approaches.

Note that since the T-REx training set includes approximately 2.2 million samples, we randomly select 5% of them to train our models to speed up experiments. It is important to note that labels for test sets of these datasets are not publicly available. Therefore, as the public validation set was not used for training or hyperparameter tuning, we use the validation set directly to evaluate our models.

4.2 Downstream RAG Models for uRAG Optimization

To have a comprehensive study, we consider a total of 18 diverse RAG models during training in our experiments. Different RAG models conduct different tasks, are trained using different resources, are using different underlying models, and/or are consuming different number of retrieved documents. Such diversity would enable us to evaluate the generalizability of different approaches. The downstream task conducted by models M_1 to M_9 is open-domain question answering. Models M_{10}, M_{11}, and M_{12} deliver fact verification functionality to their users. Finally, models M_{13} to M_{18} perform slot filling for relation extraction. All RAG models are listed in Table 2. Each of these models is trained on a different dataset or uses a different large language model. We consider three retrieval-augmented large language models:

(1) Retrieval-augmented T5 (RA-T5) is a language model based on T5-small [32] with 60 million parameters 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 ith retrieved document. Our T5 model has an input length limitation of 4096 tokens. Given this limitation, we feed $k = 10$ documents to this model. Each RA-T5 model is fine-tuned on the corresponding training set. For example, the RA-T5 model in M_1 is trained on the Natural Questions dataset variant in KILT [31] (see data statistics in Table 1).
(2) Retrieval-augmented BART (RA-BART) is BART-base [24] with 140 million parameters that uses the same augmentation approach for input format as RA-T5. BART goes under a different pre-training process than T5, thus, would exhibit a different behavior and performance. Additionally, it has an input length limitation of 1024 tokens. Therefore, we use only $k = 4$ for augmenting BART using the results returned by R_θ. Each RA-BART model is fine-tuned on the corresponding training set.
(3) Fusion-in-Decoder (FiD) [16] uses a different augmentation approach. Unlike RA-T5 and RA-BART that are 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, for which we used T5-small [32] in our experiments. We use FiD-small in our experiments. The number of augmented documents is set to $k = 10$ for FiD. Similarly, each FiD model is fine-tuned on the corresponding training set.

All retrieval-augmented models listed in Table 2 are fine-tuned separately. We use the AdamW optimizer with a weight decay of 10^{-2} and a learning rate of 5×10^{-5} for 10 epochs. Linear warmup is applied to the first 5% of training steps. The effective batch size is set to 64, achieved through gradient accumulation. Each model is trained using different computation resources, including up to 8 A100, 1080ti, and 2080ti Nvidia GPUs. We employ BM25[34], implemented in Pyserini [27], to retrieve documents for the training of the aforementioned models. To train the models, a seq2seq loss (i.e., cross-entropy between the predicted token's probability and the ground truth token distribution) is employed.

As detailed in Section 3, each RAG model is required to define a utility function that assesses its performance. To select the utility function for our RAG models, we adhere to the recommended metrics outlined by the KILT benchmark [31]. Thus, we choose the evaluation metric of each dataset as the utility function for the models that perform their task on that dataset. Table 2 illustrates the utility functions assigned to the RAG models utilized in this paper. In general, we employ Exact Match (EM) for short-form question answering datasets, ROUGE-L for long-form question answering datasets, and accuracy as the utility function for the remaining tasks. For Exact Match, we follow the post-processing approaches introduced by Rajpurkar et al. [33].

4.3 Search Engine Implementation in uRAG

We adopt a two-stage cascaded retrieval pipeline for implementing R_θ. A wikipedia dump provided by the KILT benchmark is used as the unstructured knowledge source.[4] We adhere to the preprocessing method outlined by Karpukhin et al. [20], where each document is divided into passages, each with a maximum length of 100 words. Additionally, we concatenate the document title with each passage to form the documents in the retrieval corpus. This corpus consists of approximately 36 million passages.

We use Pyserini's [27] implementation of BM25 with default parameters for implementing the first stage that takes the query string q and retrieves 100 documents. The second stage model is a cross-encoder re-ranker that takes the query and document text as input and produces a scalar as the relevance score. To "personalize" the result of the retrieval model, we contextualize the cross-encoder representation based on M_i and its downstream task T_i. Following Nogueira and Cho [30], we use a linear projection over the representation associated with the start token (i.e., `[CLS]`) to obtain the relevance score, as follows:

$$s_d = \textsc{Encoder}_{[\text{CLS}]}(\texttt{tid}; \texttt{mid}; q; d) \cdot W \tag{3}$$

where `tid` and `mid` represent the downstream task and model identifiers, respectively. The sign ; denotes concatenation with a separation token. $W \in \mathbb{R}^{D\times 1}$ is a linear projection layer, where D represents the embedding dimensionality of the encoder. In this paper, we utilize the BERT-base [7] as the text encoder. Once all the 100 documents are re-ranked, the top k documents are selected as the model's output, denoted as L_q. Here, the search engine logs the query and the returned results for later use, e.g., using them for optimization when the feedback from the RAG model is received.

For optimization, R_θ solicits feedback one by one for each document $d \in L_q$. We then apply a threshold $\tau_{(\text{tid,mid})}$ to the feedback U_i received for each document; the feedback higher than or equal to the threshold is considered as a positive feedback, negative otherwise. Based on these feedbacks, we create a set of positive (D^q_{pos}) and negative (D^q_{neg}) documents for each query. We then apply a binary cross-entropy loss function to train the cross-encoder model:

$$L = -\frac{1}{|Q|}\sum_{q\in Q}\Big(\sum_{d\in D^q_{\text{pos}}} \log\sigma(s_d) + \sum_{d\in D^q_{\text{neg}}} \log\sigma(1-s_d)\Big) \tag{4}$$

where Q is the set of all queries used for training. Note that we randomly substitute the model ID and task ID with the *"unk"* token for 10% of training samples. This aids the model in adapting to new tasks and models during inference. The rationale behind this approach is to train the model to correctly assign labels to samples even when it lacks information about the specific task and model being performed. For training R_θ, we use Adam optimizer [21] with a learning rate of 10^{-5} for two epochs. Linear warmup is applied to the first 5% of training steps. The effective batch size is set to 512, which is achieved through gradient accumulation. Values of $k = 32$ and $\tau_{(\text{tid,mid})} = 0.5$ are consistently employed in all experiments. The maximum input length for this model is set to 256 tokens.

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

5 EMPIRICAL RESULTS AND FINDINGS

This section is dedicated to presenting supporting results that address the research questions outlined in Section 1.

RQ1: How does unified reranking perform compared to training individual rerankers for each RAG model? The results presented in Table 3 offer insights into answering this research question. It is evident all reranking models applied to BM25 lead to improved downstream performance for all 18 models we have in our experiments. The only exception is the models trained and evaluated on FEVER dataset for fact verification. As a point of reference, we also compare the results with Contriever [14]–an effective zero-shot dense retrieval models. Contriever consistently underperforms BM25 for all downstream models M_1 to M_{18}.

Additionally, comparing the results obtained by $\text{Reranker}_{\text{unified}}$ and $\text{Reranker}_{\text{individual}}$ in Table 3 shows that developing a unified retrieval models for all retrieval-augmented models achieves a better performance for 78% of models (i.e., 14 out of 18 models; all except M_8 on HotPotQA and M_{16} to M_{18} on T-REx). For the four models that unified retrieval does not show improvement, we do not observe statistically significant differences between model performances. Thus, we can conclude unified retrieval performs on par or significantly better than individual retrieval optimization in almost all cases. Statistically significant improvement has been observed in 11 out of 18 downstream models. Looking at the overall performance, we show that unified retriever leads to statistically significantly higher performance compared to individual retrievers.

In conclusion, the discussion above suggests that our approach, which leverages the feedback from all RAG models across all tasks to optimize the search engine, is significantly superior to the alternative of training a single retrieval model for each RAG model, which is the current standard in developing RAG systems.

RQ2: How does unified reranking perform compared to training rerankers using the feedback obtained from all RAG systems that use the same dataset? Instead of retrieval optimization for individual models, one may suggest to train a retrieval model for the downstream models that are using the same task and dataset, which we call $\text{Reranker}_{\text{dataset}}$. Comparing the results obtained by this approach and $\text{Reranker}_{\text{unified}}$ in Table 3 suggest that training a unified reranker across the feedback of all users on average statistically significantly outperforms the reranker trained on the feedbacks of models performing on the same dataset. In more details, unified optimization performs better for 72% of models (i.e., 13 out of 18 models; all models except M_{10}, M_{11}, M_{12}, M_{17}, and M_{18}, that are all performing on either FEVER or T-REx datasets). Note that there is statistically significant differences in M_{11} and M_{12} users. However, in other cases, there is not a statistically significant difference between the results for M_{10}, M_{17}, and M_{18}. Considering overall performance across models, we observe statistically significant improvements when using unified training.

In summary, the observations above indicate that for all downstream RAG models, leveraging feedback from all RAG models across tasks for optimizing the reranking model is either on par or significantly superior to the alternative approach of training a retrieval model for each dataset. Therefore, retrieval models can benefit from knowledge transfer across datasets and tasks for RAG.

Table 3: Downstream performance obtained by 18 RAG models listed in Table 2 consuming different retrieval results. The overal performance is the macro-average of the performance of 18 RAG models. Superscripts 1, 2, 3, 4, and 5 denote statistically significant improvements in the performance compared to BM25, Contriever, $\text{Reranker}_{\text{individual}}$, $\text{Reranker}_{\text{dataset}}$, and $\text{Reranker}_{\text{unified}}$ w/o personalization, respectively. We used McNemar statistical test for significant test ($p < 0.05$).

RAG Model	Data & Metric	BM25	Contriever	$\text{Reranker}_{\text{individual}}$	$\text{Reranker}_{\text{dataset}}$	$\text{Reranker}_{\text{unified}}$ w/o personalization	$\text{Reranker}_{\text{unified}}$
M_1	NQ - EM	28.05	22.55	36.76	37.39	37.18	37.82^{12}
M_2	NQ - EM	33.09	23.68	40.07	40.50	42.29	42.12^{1234}
M_3	NQ - EM	29.64	23.69	40.50	41.14	40.47	42.37^{12345}
M_4	TriviaQA - EM	51.35	44.33	59.28	60.25	60.30	60.68^{123}
M_5	TriviaQA - EM	57.52	48.49	64.76	67.23	67.30	68.12^{123}
M_6	TriviaQA - EM	60.48	49.30	67.44	68.63	69.14	68.74^{123}
M_7	HotPotQA - EM	27.51	18.80	29.92	30.91	31.32	31.33^{123}
M_8	HotPotQA - EM	31.21	20.78	35.03	34.62	34.10	34.85^{124}
M_9	HotPotQA - EM	29.48	20.43	32.54	32.71	32.96	33.46^{1234}
M_{10}	FEVER - Accuracy	86.83	84.21	86.24	86.83	87.02	86.46^{2}
M_{11}	FEVER - Accuracy	87.54	84.37	84.38	87.54	86.78	85.99^{23}
M_{12}	FEVER - Accuracy	87.04	86.74	86.02	87.04	87.21	86.55^{23}
M_{13}	zsRE - Accuracy	55.37	38.77	60.39	59.98	61.41	61.09^{124}
M_{14}	zsRE - Accuracy	51.42	29.05	59.29	58.96	60.92	60.58^{1234}
M_{15}	zsRE - Accuracy	55.42	37.35	60.47	60.66	62.08	62.13^{1234}
M_{16}	T-REx - Accuracy	70.88	56.94	73.58	72.86	73.70	72.92^{12}
M_{17}	T-REx - Accuracy	75.16	58.30	80.04	80.18	79.64	79.94^{12}
M_{18}	T-REx - Accuracy	78.88	65.06	80.78	80.34	80.86	80.24^{12}
Overall	–	55.38	45.15	59.86	60.43	60.81	60.85^{1234}

RQ3: How does "personalizing" search results for different RAG systems impact the performance? To assess the impact of personalization on the search engine for each RAG model, an experiment was conducted where a reranker was trained similarly to unified reranker but without incorporating task and model IDs as the input of reranker. The results of this experiment are illustrated in Table 3. The results suggest that personalization leads to an improvement on average across all users, but it is not significant.

More specifically, the improvements resulting from personalization are most pronounced in the question answering tasks. This is attributed to the similarity among most questions in these datasets, making it challenging for the non-personalized model to discern the specific task associated with a given query. Hence, personalization aids the reranker in identifying the task for which the input is intended and subsequently reranking the documents accordingly. Conversely, the improvements in the slot filling tasks (i.e., T-REx and zsRE) are smaller and for fewer users. Additionally, personalization adversely affects performance in the FEVER dataset. These observations indicate that while, on average, our approach for personalization was effective, there is potential for a further study on how to effectively personalize the search engine for RAG models, which is an important topic for future exploration.

RQ4: How does unified reranking perform for new (unknown) RAG systems that use an observed (known) dataset? We design an experiment in which we incorporate three new RAG models with different architecture and pre-training process:

(1) Retrieval-augmented PEGASUS (RA-PEGASUS) uses in-prompt augmentation with the PEGASUS language model [58] which has 570 million parameters. We use the same input format as used by RA-T5 and RA-BART (see Section 4.2). PEGASUS has a 1024 maximum token limit as its input; thus we limit this model to consume $k = 4$ retrieved documents. RA-PEGASUS models are also fined-tuned on the training set of the corresponding datasets. The fine-tuning details are the same as other retrieval-augmented models presented in Section 4.2.
(2) Retrieval-augmented Mistral (RA-Mistral) also uses in-prompt augmentation for Mistral LLM with 7 billion parameters [18],

Table 4: A list of RAG models used in our experiments for evaluating generalizability of uRAG. ELI5 is also an open-domain QA dataset, but the expected outputs are longer (e.g., sentences or a paragraph) compared to other datasets.

	Task	Data	Utility Func.	LM
M_{19}	entity linking	AY2	Accuracy	RA-T5
M_{20}	long-form QA	ELI5	ROUGE-L	RA-T5
M_{21}	open-domain QA	SQuAD	Exact Match	RA-T5
M_{22}	entity linking	AY2	Accuracy	RA-BART
M_{23}	long-form QA	ELI5	ROUGE-L	RA-BART
M_{24}	open-domain QA	SQuAD	Exact Match	RA-BART
M_{25}	open-domain QA	NQ	Exact Match	RA-PEGASUS
M_{26}	open-domain QA	TriviaQA	Exact Match	RA-PEGASUS
M_{27}	long-form QA	ELI5	ROUGE-L	RA-PEGASUS
M_{28}	open-domain QA	SQuAD	Exact Match	RA-PEGASUS
M_{29}	open-domain QA	NQ	Exact Match	RA-Mistral
M_{30}	open-domain QA	TriviaQA	Exact Match	RA-Mistral
M_{31}	long-form QA	ELI5	ROUGE-L	RA-Mistral
M_{32}	open-domain QA	SQuAD	Exact Match	RA-Mistral
M_{33}	open-domain QA	NQ	Exact Match	RA-Llama2
M_{34}	open-domain QA	TriviaQA	Exact Match	RA-Llama2
M_{35}	long-form QA	ELI5	ROUGE-L	RA-Llama2
M_{36}	open-domain QA	SQuAD	Exact Match	RA-Llama2

Table 5: The performance on downstream task obtained for new (unknown) models applied to the datasets known to R_θ. Superscript [1] and [2] denote statistically difference between our model and BM25 and Reranker$_{\text{dataset}}$, respectively ($p < 0.05$).

RAG Model	Data & Metric	BM25	Reranker$_{\text{dataset}}$	Reranker$_{\text{unified}}$
M_{25}	NQ - EM	33.38	45.08	**45.29**[1]
M_{29}	NQ - EM	15.15	**21.36**	21.25[1]
M_{33}	NQ - EM	11.13	16.53	**16.56**[1]
M_{26}	TQA - EM	53.53	61.74	**62.39**[1]
M_{30}	TQA - EM	44.74	**50.40**	50.19[1]
M_{34}	TQA - EM	25.86	**29.63**	29.22[1]

with no fine-tuning. Instead, to increase the diversity of models in our experiments, we use few-shot in-context learning for this approach, where five training examples are presented to the model in its prompt. Mistral can accept up to 32k tokens as input (including the few-shot examples). We feed $k = 5$ retrieved documents to this model.[5]

(3) Retrieval-augmented Llama2 (RA-Llama2) uses an in-prompt augmentation approach similar to RA-Mistral, but using Llama2 with 13 billion parameters[47]. It also uses few-shot in-content learning, but with two examples instead (due to the input length limitation). Llama2 can accept up to 4096 tokens as input. RA-Llama2 uses $k = 3$ retrieved documents.[6]

To address this research question, we evaluate these models on two existing datasets from our pipeline: Natural Questions (NQ) and TriviaQA (TQA). These models are denoted as M_{25}, M_{26}, M_{29}, M_{30}, M_{33}, and M_{34}, as is presented in Table 4. Since these models are new to our rerankers, their model and task ID is set to unknown, thus no personalization is conducted for these models.

By comparing our unified reranking model with BM25 without reranking in Table 5, we observe significant improvements in all cases in comparison with BM25. The improvements range from 11% to about 50%. Larger improvements are observed for the models evaluated on the Natural Questions dataset, which was also the case for the models used in our training process (see Table 3). Moreover, an alternative approach involves leveraging Reranker$_{\text{dataset}}$, which has been trained using the feedbacks from multiple large language models. The results of this alternative approach are also presented in Table 5. The results reveal that, for certain users, Reranker$_{\text{unified}}$ yields superior results, while for others, Reranker$_{\text{dataset}}$ performs better. Additionally, there is no statistically significant difference in performance between these models, suggesting comparable effectiveness in leveraging feedback from multiple large language models when it comes to serving a new large language model.

RQ5: How does unified reranking perform for a known RAG system on a new (unknown) dataset that is similar to the ones used during training? To study this question, we conduct an experiment in which RA-BART and RA-T5 are employed as known models to the search engine for a new dataset, SQuAD [33], which performs a known task (i.e., short-form open-domain question answering). These models are encoded as M_{21} and M_{24} (see Table 4). We use the open-domain variant of SQuAD [33] where the gold passage associated to each question is unknown. The results of this experiment are presented in Table 6. The results suggest that when the new task is similar to the known tasks for the search engine, as in the case of SQuAD being similar to short-form question answering datasets used during training (i.e., NQ, TQA), significant improvements can still be achieved in comparison with the baseline (i.e., BM25). Note that the selection of baselines in this experiment is influenced by the consideration that other reranker baselines listed in Table 3 are customized for specific model or dataset; they are not applicable to new models and datasets.

Table 6: The performance for new (unknown) models applied to the new datasets. SQuAD is a question answering dataset with short answers, thus similar to the datasets used during training such as NQ and TriviaQA. ELI5 focuses on long-form question answering, a task fundamentally different from datasets used during R_θ training. AY2 is a dataset for entity linking, which is not used for training R_θ. Superscript * denotes significant improvement over BM25 ($p < 0.05$).

RAG Model	Data & Metric	BM25	Reranker$_{\text{unified}}$
M_{21}	SQuAD - EM	35.91	**41.19***
M_{24}	SQuAD - EM	32.92	**36.70***
M_{19}	AY2 - Accuracy	**75.52**	73.3
M_{22}	AY2 - Accuracy	81.98	**82.08**
M_{20}	ELI5 - ROUGE-L	15.40	**15.42**
M_{23}	ELI5 - ROUGE-L	**23.42**	23.24

Table 7: The performance for new (unknown) models applied to new datasets that are unknown to R_θ. Superscript * denotes statistically significant improvement over BM25 ($p < 0.05$).

RAG Model	Data & Metric	BM25	Reranker$_{\text{unified}}$
M_{28}	SQuAD - EM	36.37	**42.04***
M_{32}	SQuAD - EM	13.01	**15.18***
M_{36}	SQuAD - EM	10.31	**12.06***
M_{27}	ELI5 - ROUGE-L	**19.84**	19.79
M_{31}	ELI5 - ROUGE-L	**22.03**	21.98
M_{35}	ELI5 - ROUGE-L	**19.16**	18.77

RQ6: How does unified reranking perform for a known RAG system on entirely new tasks? To study this question, we conduct an experiment in which RA-BART and RA-T5 are employed as known models to the search engine for new tasks. ELI5 [10] serves as a long-form question answering task and AY2 [13] as an entity linking dataset, which are new tasks for the search engine. These models are denoted as M_{19}, M_{20}, M_{22}, and M_{23} (see Table 4). The results of this experiment, presented in Table 6, suggest that when tasks are considerably different from the ones observed during training, unified retrieval as implemented in this approach is not likely to help. Indeed, unified retrieval model demonstrates superior performance for some users; however, there is no statistical

[5] RA-Mistral uses the following prompt: `<s>[INST] you are a helpful assistant. following the answer patterns in the provided examples, answer the question concisely using the hints without any explanation. {shot$_1$} ... {shot$_m$} [/INST] question: {input}? hint: {context$_1$} ... {context$_k$} answer:`

[6] RA-Llama2 uses the following prompt: `you are a helpful assistant. following the answer patterns in the provided examples, answer the question concisely using the hints without any explanation. {shot$_1$} ... {shot$_m$} question: {input}? hint: {context$_1$} ... {context$_k$} answer:`

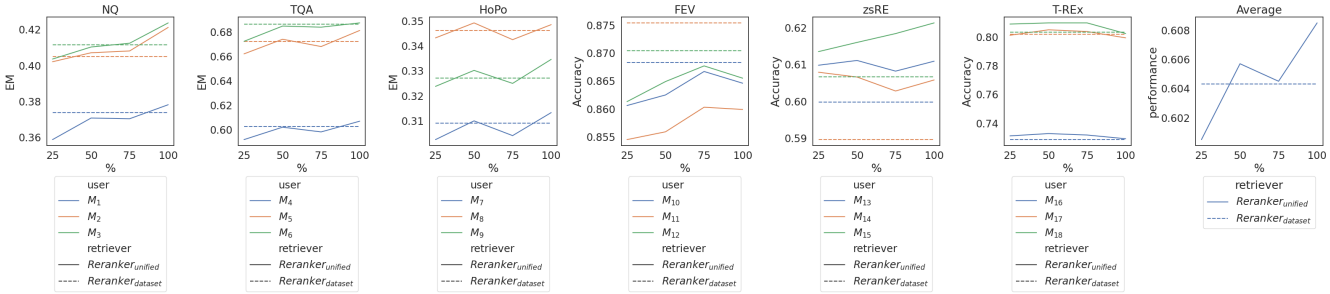

Figure 3: The performance of unified retrieval model using different percentages of training data. The dashed line indicates that the model is trained on the full dataset. The far right plot demonstrate the overall average performance across all datasets.

difference between the unified retrieval model and the baseline. This creates an important opportunity for future work to focus on generalizability and adaptability of uRAG to unknown tasks.

RQ7: How does unified reranking perform for a new (unknown) RAG system on a new (unknown) dataset that is similar to the ones used during training? To answer this question, in this experiment, we utilize RA-PEGASUS, RA-Mistral, and RA-Llama2 on SQuAD. These are models that did not exist at the time of training R_θ and SQuAD is a new (unknown) dataset for R_θ but related to NQ and TriviaQA. These models are encoded as M_{28}, M_{32}, and M_{36} (see Table 4). The results of this experiment are displayed in Table 7. The findings indicate that when the task is similar to those encountered during model training (e.g., SQuAD in this case similar to NQ and TriviaQA in the training tasks), significant improvements over the baseline can be achieved even though the RAG model is unknown to the search engine.

RQ8: How does unified reranking perform for a new RAG system on an entirely new (unknown) task? In the most extreme case, both the task and the model are unknown to the search engine. Here, we utilize RA-PEGASUS, RA-Mistral, and RA-Llama2 on ELI5 dataset as a new task that is relatively different from the tasks used during training. These models are encoded as M_{27}, M_{31}, and M_{35}. The results in Table 7 indicate that when the task is substantially different from those available during training, as is in ELI5 (i.e., long-form question answering), there is a drop in performance, though not statistically significant.

RQ9: How does uRAG perform when different amount of training data is provided? We vary the amount of training data obtained from each downstream RAG model to 25%, 50%, 75%, and 100% of the data used in other experiments. The results in Figure 3 show that even using 25% of the training data, our approach is sufficiently effective to outperform the baseline trained on the full datasets (i.e., $\text{Reranker}_{\text{dataset}}$) in average. This trend continues when our approach consumes more data where it outperforms the baseline significantly when trained on the full datasets.

Furthermore, for most users, augmenting the training data improves the performance. Particularly noteworthy is the fact that for the FEVER and T-REx datasets, the optimal performance is achieved when utilizing 75% of the training data. Importantly, for the majority of users, excluding M_1, M_6, M_{10}, M_{11}, and M_{12}, our approach either outperforms or performs equivalently to the baseline, trained on the full datasets, by observing 50% of the training data. This underscores the effectiveness of unified reranking, especially when confronted with limited training data, for most users.

6 CONCLUSION & FUTURE WORK

We studied the promises and challenges in developing a search engine for multiple downstream RAG systems. We introduced the uRAG ecosystem that enabled us to conduct large-scale experimentation to answer some fundamental research questions in this area. We showed that optimizing a unified reranker leads to significant improvements compared to the current standard recipe–i.e., learning a retrieval model per downstream RAG model. We highlighted the improvements observed by the unified reranker is generalizable to new (unknown) models that are based on known datasets or unknown datasets that are closely related to the ones used in the training procedure. We further explained the challenges in personalizing rerankers for downstream RAG models and generalizing to unknown downstream tasks.

This paper opens up a wide range of future research directions. Through the proposed uRAG ecosystem, future work can focus on (1) optimal aggregation of feedback from various downstream RAG models, (2) calibrating their feedback, (3) considering multiple utility functions per downstream model, (4) expanding the work to retrieval model optimization instead of reranking, (5) studying novel regularization techniques for improving uRAG generalization, (6) studying online optimization of methods for uRAG as downstream RAG models interact with the retrieval model, (7) exploring interleaving and counterfactual learning approaches in the context of uRAG, (8) going beyond text generation and extending to a more general REML scenario, and many more.

ACKNOWLEDGMENTS

The authors would like to thank Fernando Diaz for invaluable discussions. This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant number 2143434, in part by the Office of Naval Research contract number N000142212688, in part by Lowe's, and in part by an Amazon Research Award, Fall 2022 CFP. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] 2024. The ChatGPT Retrieval Plugin. https://github.com/openai/chatgpt-retrieval-plugin. Accessed: 2024-01-20.

[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. 2023. Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection. arXiv:2310.11511 [cs.CL]

[4] Deng Cai, Yan Wang, Huayang Li, Wai Lam, and Lemao Liu. 2021. Neural Machine Translation with Monolingual Translation Memory. 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, 7307–7318. https://doi.org/10.18653/v1/2021.acl-long.567

[5] 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

[6] Hao Cheng, Yelong Shen, Xiaodong Liu, Pengcheng He, Weizhu Chen, and Jianfeng Gao. 2021. UnitedQA: A Hybrid Approach 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, 3080–3090. https://doi.org/10.18653/v1/2021.acl-long.240

[7] 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

[8] Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of Wikipedia: Knowledge-Powered Conversational Agents. In *International Conference on Learning Representations*. https://openreview.net/forum?id=r1l73iRqKm

[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] Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. ELI5: Long Form Question Answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 3558–3567. https://doi.org/10.18653/v1/P19-1346

[11] Michael Glass, Gaetano Rossiello, Md Faisal Mahbub Chowdhury, Ankita Naik, Pengshan Cai, and Alfio Gliozzo. 2022. Re2G: Retrieve, Rerank, Generate. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz (Eds.). Association for Computational Linguistics, Seattle, United States, 2701–2715. https://doi.org/10.18653/v1/2022.naacl-main.194

[12] 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

[13] Johannes Hoffart, Mohamed Amir Yosef, Ilaria Bordino, Hagen Fürstenau, Manfred Pinkal, Marc Spaniol, Bilyana Taneva, Stefan Thater, and Gerhard Weikum. 2011. Robust Disambiguation of Named Entities in Text. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, Regina Barzilay and Mark Johnson (Eds.). Association for Computational Linguistics, Edinburgh, Scotland, UK., 782–792. https://aclanthology.org/D11-1072

[14] 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

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

[16] 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

[17] 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

[18] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7B. arXiv:2310.06825 [cs.CL]

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

[22] 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

[23] 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

[24] 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

[25] 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.

[26] Huayang Li, Yixuan Su, Deng Cai, Yan Wang, and Lemao Liu. 2022. A Survey on Retrieval-Augmented Text Generation. arXiv:2202.01110 [cs.CL]

[27] Jimmy Lin, Xueguang Ma, Sheng-Chieh Lin, Jheng-Hong Yang, Ronak Pradeep, and Rodrigo Nogueira. 2021. Pyserini: A Python Toolkit for Reproducible Information Retrieval Research with Sparse and Dense Representations. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (, Virtual Event, Canada,) *(SIGIR '21)*. Association for Computing Machinery, New York, NY, USA, 2356–2362. https://doi.org/10.1145/3404835.3463238

[28] 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

[29] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. WebGPT: Browser-assisted question-answering with human feedback. arXiv:2112.09332 [cs.CL]

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

[31] 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

[32] 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.

[33] 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

[34] 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

[35] 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

[36] 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

[37] Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024. Optimization Methods for Personalizing Large Language Models through Retrieval Augmentation. In *Proceedings of the 47th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington, DC, USA) *(SIGIR '24)*. (to appear).

[38] Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024. LaMP: When Large Language Models Meet Personalization. arXiv:2304.11406 [cs.CL]

[39] Alireza Salemi, Mahta Rafiee, and Hamed Zamani. 2023. Pre-Training Multi-Modal Dense Retrievers for Outside-Knowledge Visual Question Answering. arXiv:2306.16478 [cs.IR]

[40] Alireza Salemi and Hamed Zamani. 2024. Evaluating Retrieval Quality in Retrieval-Augmented Generation. In *Proceedings of the 47th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington, DC, USA) *(SIGIR '24)*. (to appear).

[41] 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

[42] Xuehua Shen, Bin Tan, and ChengXiang Zhai. 2005. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Salvador, Brazil) *(SIGIR '05)*. Association for Computing Machinery, New York, NY, USA, 43–50. https://doi.org/10.1145/1076034.1076045

[43] 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

[44] 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

[45] Yiping Song, Cheng-Te Li, Jian-Yun Nie, Ming Zhang, Dongyan Zhao, and Rui Yan. 2018. An Ensemble of Retrieval-Based and Generation-Based Human-Computer Conversation Systems. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*. International Joint Conferences on Artificial Intelligence Organization, 4382–4388. https://doi.org/10.24963/ijcai.2018/609

[46] 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

[47] Hugo Touvron et al. 2023. Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs.CL]

[48] Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the Planning Abilities of Large Language Models - A Critical Investigation. *arXiv 2305.15771* (2023).

[49] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc Le, and Thang Luong. 2023. FreshLLMs: Refreshing Large Language Models with Search Engine Augmentation. In *arXiv*.

[50] 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.

[51] 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

[52] Dingmin Wang, Qiuyuan Huang, Matthew Jackson, and Jianfeng Gao. 2023. NLP Agent - Retrieve What You Need: A Mutual Learning Framework for Open-domain Question Answering. *Proceedings of Transactions of the Association for Computational Linguistics (TACL)* (May 2023). https://www.microsoft.com/en-us/research/publication/retrieve-what-you-need-a-mutual-learning-framework-for-open-domain-question-answering/

[53] Jason Weston, Emily Dinan, and Alexander Miller. 2018. Retrieve and Refine: Improved Sequence Generation Models For Dialogue. In *Proceedings of the 2018 EMNLP Workshop SCAI: The 2nd International Workshop on Search-Oriented Conversational AI*. Association for Computational Linguistics, Brussels, Belgium, 87–92. https://doi.org/10.18653/v1/W18-5713

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

[55] 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

[56] Hamed Zamani and Michael Bendersky. 2024. Stochastic RAG: End-to-End Retrieval-Augmented Generation through Expected Utility Maximization. In *Proceedings of the 47th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington, DC, USA) *(SIGIR '24)*. (to appear).

[57] 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

[58] Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020. PEGASUS: Pre-Training with Extracted Gap-Sentences for Abstractive Summarization. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 1051, 12 pages.

[59] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jian-Yun Nie, and Ji-Rong Wen. 2023. A Survey of Large Language Models. arXiv:2303.18223 [cs.CL]

[60] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. 2021. Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. arXiv:2101.00774 [cs.AI]