

LURE-RAG: Lightweight Utility-driven Reranking for Efficient RAG

Manish Chandra¹, Debasis Ganguly¹, and Iadh Ounis¹

University of Glasgow, Glasgow, United Kingdom

m.chandra.1@research.gla.ac.uk, Debasis.Ganguly@glasgow.ac.uk,
iadh.ounis@glasgow.ac.uk

Abstract. Most conventional Retrieval-Augmented Generation (RAG) pipelines rely on relevance-based retrieval, which often misaligns with utility – that is, whether the retrieved passages actually improve the quality of the generated text specific to a downstream task such as question answering or query-based summarization. The limitations of existing utility-driven retrieval approaches for RAG are that, firstly, they are resource-intensive typically requiring query encoding, and that secondly, they do not involve listwise ranking loss during training. The latter limitation is particularly critical, as the relative order between documents directly affects generation in RAG. To address this gap, we propose Lightweight Utility-driven Reranking for Efficient RAG (LURE-RAG), a framework that augments any black-box retriever with an efficient LambdaMART-based reranker. Unlike prior methods, LURE-RAG trains the reranker with a listwise ranking loss guided by LLM utility, thereby directly optimizing the ordering of retrieved documents. Experiments on two standard datasets demonstrate that LURE-RAG achieves competitive performance, reaching 97–98% of the state-of-the-art dense neural baseline, while remaining efficient in both training and inference. Moreover, its dense variant, UR-RAG, significantly outperforms the best existing baseline by up to 3%.

Keywords: RAG, listwise ranking, lightweight reranker

1 Introduction

Large Language Models (LLMs) have shown strong ability to generate fluent and often factually grounded text, yet they are limited by their parametric knowledge (which is fixed at training time) [28,4]. They may also hallucinate or fail on domain-specific or newly emerging information. Retrieval-Augmented Generation (RAG) has emerged as a prominent framework to address these limitations by combining external retrieval of relevant documents with language model generation [38,12]. RAG systems first retrieve documents or passages from some corpus given an input, then augment the LLM’s input with those documents, and finally generate a response grounded on both retrieved and internal knowledge. RAG has become a core paradigm in knowledge-intensive NLP tasks [18,14].

RAG systems usually consist of multiple modules including at least a retriever and a generator. Some systems may have other modules to further enhance effectiveness on downstream tasks, like a reranker [13] or a decision maker deciding when to retrieve [22,11].

In many RAG pipelines [4,12], relevance is defined in the traditional IR sense, such as lexical or semantic similarity between documents and the query. However, in the RAG setting, semantic relevance to the query alone does not necessarily translate into utility for the LLM’s downstream generation [40,9,42], that is, whether the retrieved documents actually help the model produce more accurate, coherent, or useful answers. This discrepancy arises because, even when documents are individually relevant, concatenating them into a single context may introduce incoherence or inconsistency, which can in turn mislead the LLM and degrade the quality of the generated output. Moreover, even if a document is semantically similar to an input question, it may miss crucial facts required for the correct answer generation. Recent works [11,38] showed that using utility-driven signals (e.g. metrics derived from LLM outputs and the ground-truth answers) as supervision can better align retrieval (and subsequent reranking in some cases) to what improves generation.

Despite the progress in the direction of utility-driven signals, there remain important gaps. Firstly, the loss functions used in the retriever or reranker training do not reflect order or comparative ranking (i.e., a document in rank 1 vs rank 3 matters). This formulation treats all mis-rankings equally, failing to reflect the importance of ranking errors at the top of the list versus those deeper in the ranking. It has indeed been shown that the order of documents matter in RAG [31,9]. Secondly, the existing utility-driven methods are not efficient in terms of training and inference cost. To the best of our knowledge, there is no existing work focusing on lightweight, efficient reranking methods that still integrate utility supervision in a RAG setup.

In this work, we propose a framework that treats the retriever and the LLM as black-boxes, and adds a lightweight reranker on top of the retrieved candidates, which is efficient both for training and inference. We train the reranker using listwise ranking loss derived from supervision given by the LLM utility, i.e., how much each document/passage contributes to the quality of generation. We call our proposed framework **LURE-RAG** (Lightweight Utility-driven Reranking for Efficient RAG).

Our Contributions. Our main contributions are summarized below.

- We propose a *black-box retriever, lightweight reranker and black-box generator* architecture for RAG, called LURE-RAG.
- We propose a listwise ranking-loss based training scheme for the reranker that uses LLM utility signals (answer correctness) to supervise which retrieved documents should be ranked higher.
- We demonstrate that LURE-RAG shows a competitive performance, coming within 97-98% of the strongest dense neural reranker baseline, while being computationally efficient with modest training and inference overhead, making it practical for real-world RAG systems.

- We demonstrate that the dense neural counterpart of LURE-RAG, which we call UR-RAG (Utility-driven Reranking for RAG), significantly outperforms competing baselines on multiple knowledge-intensive tasks.

2 Related Work

In this section, we position LURE-RAG within the literature on RAG, distinguishing between those methods that do not explicitly use utility-driven supervision, and those that do. We then cover learning to rank using listwise or pairwise ranking losses, since LURE-RAG leverages those approaches.

2.1 Retrieval Augmented Generation

RAG [28] refers broadly to systems which combine a retrieval component (obtaining a ranked list of the most likely relevant documents from a collection) together with a generative (LLM) component that uses those retrieved passages to output answers. RAG helps address LLM limitations like outdated knowledge, hallucinations, or inability to access domain-specific or emerging information [28,12].

Relevance-based RAG. Early approaches to RAG rely primarily on the traditional IR notion of relevance, rather than explicitly optimizing for downstream LLM utility [4,12]. In these methods, retrieval is guided by query-document lexical or semantic similarity, often using sparse methods such as BM25 [37] or dense dual-encoder retrievers [25]. Once retrieved, the top- k passages are either directly provided to the generator [28,8], or fused using models such as Fusion-in-Decoder (FiD) [20].

Supervised Retriever (Utility-driven RAG). These methods incorporate a downstream LLM output utility (e.g., the correctness of answers in a factoid QA task) into the training of retriever or reranker. A noticeable example is RePlug [38], which integrates a dense retriever (e.g., Contriever [21]) into a black-box LLM pipeline. Unlike classical supervised retrievers trained on relevance labels, RePlug leverages LLM utility signals, i.e., how much a retrieved passage improves the LLM’s likelihood of producing the correct answer. This methodology aligns the retriever with the needs of the generator rather than with human-annotated relevance judgments. The training objective in RePlug relies on KL divergence between the retriever’s predicted distribution over passages and the LLM-utility-derived distribution.

While RePlug is described as a plug-in approach, in practice, it depends on dense retrievers such as Contriever. This makes it incompatible with lightweight or non-parametric retrievers such as BM25, because BM25 does not produce differentiable representations that can be trained against utility-derived soft labels. As a result, RePlug cannot directly optimize BM25 or other non-parametric

retrievers, limiting its generality in resource-constrained or latency-sensitive settings. Furthermore, the KL divergence objective in RePlug does not take into account the utility-driven ranking of the documents, i.e., it treats misclassifications at the top of the list the same as those at the bottom. In contrast, our proposed method LURE-RAG makes use of a ranking loss and is compatible with any retriever since it assumes the retriever to be a black-box.

Another recent work [42] highlights the misalignment between topical relevance and actual utility in RAG, proposing that LLMs can serve as zero-shot utility judges through a k-sampling listwise approach to mitigate inherent position biases. While effective, this method is computationally expensive at inference time, as it requires multiple LLM calls to shuffle and aggregate document scores. In contrast, our proposed LURE-RAG framework distills this utility-driven intelligence into a lightweight LambdaMART-based reranker, providing an efficient and scalable way to integrate utility signals without requiring multiple LLM calls. We do not adopt this approach as a baseline in our experiments since its reliance on iterative LLM calls makes it computationally prohibitive at scale, and its design (direct utility judgment rather than reranking) targets a different problem formulation than ours.

2.2 Lightweight Learning to Rank

Learning to Rank (LTR) in IR focuses on training models to order documents by relevance. Among the most influential approaches are RankNet [5], LambdaRank [6], and LambdaMART [7].

RankNet introduced a neural network-based pairwise ranking model that learns by minimizing a cross-entropy loss between predicted pairwise preferences and ground-truth labels. RankNet demonstrated the effectiveness of using gradient-based optimization for ranking, a key contribution for subsequent neural ranking models such as [34,2]. LambdaRank improved upon RankNet by introducing the concept of lambdas: gradient signals that depend not only on the pairwise preference but also on the impact of swapping two documents on an evaluation metric such as NDCG (Normalized Discounted Cumulative Gain). This allowed models to optimize ranking metrics more directly without needing their gradients explicitly.

LambdaMART combined LambdaRank with MART (Multiple Additive Regression Trees) [10], a boosted tree ensemble method. By applying the lambda-based gradients to tree-based learners, LambdaMART achieved state-of-the-art results in large-scale learning-to-rank benchmarks and became widely deployed in industrial search [7,41] and recommendation systems [17]. LambdaMART is particularly appealing due to its efficiency, scalability, and interpretability, which make it suitable as a lightweight reranker in real-world pipelines. In our work, we build on these strengths by employing LambdaMART within LURE-RAG as the core reranker, using LLM-derived utility signals to provide supervision. This allows us to capture task-specific document preferences while retaining the efficiency and transparency of tree-based models.

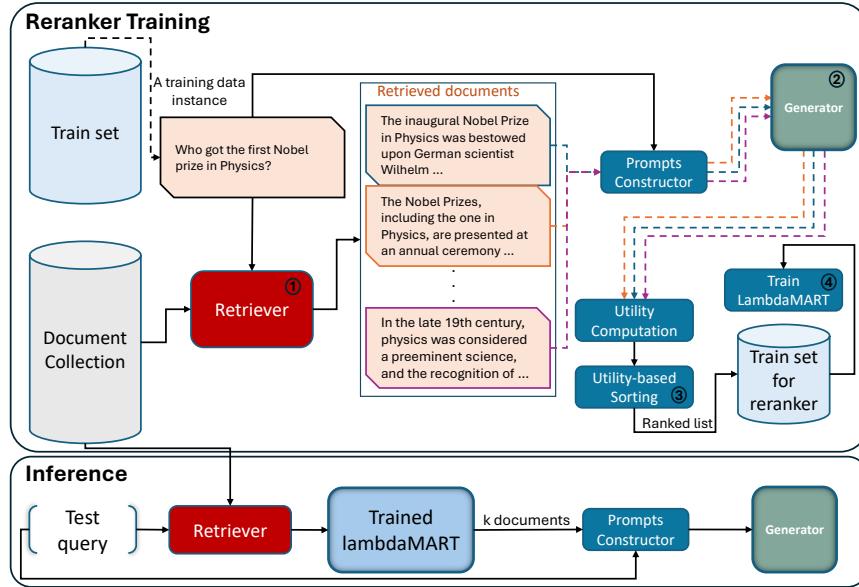


Fig. 1: Schematic diagram of LURE-RAG workflow. ① For the given query, documents are retrieved using a (black-box) retriever. ② Each of these documents is used as a context for the query and the prompts are constructed. These prompts are fed into a generator one by one to get the LLM predictions. ③ LLM’s posteriors are used to compute the utility of each document. The documents are then sorted based on the utility scores and only top- k of them are retained. These form the training instances for training a reranker. ④ LambdaMART is trained using thus obtained training data.

3 Proposed Methodology

In this section, we first provide the problem definition and then the details of our proposed strategy for lightweight utility-driven reranking for efficient RAG (LURE-RAG). Figure 1 presents an overarching view of the proposed approach.

3.1 Problem Definition

Let \mathcal{Q} denote a set of queries (or user questions), where each query $q \in \mathcal{Q}$ is represented as a sequence of tokens $q = (w_1, \dots, w_{|q|})$. For a given query q , a retriever \mathcal{R} returns a candidate set of N documents: $D_q = \{d_1, \dots, d_N\}, d_i \in \mathcal{D}$, where \mathcal{D} is the document collection. In a traditional retrieval setting, documents are ranked according to relevance scores. However, in RAG, what ultimately matters is the *utility* of a document, i.e., how much it improves the LLM’s ability to generate correct responses.

We formalize the utility-driven reranking problem as follows: given a query q , a set of candidate documents D_q , and a utility function $U(q, d)$ estimated via

LLM outputs, learn a ranking function: $f_\theta : (q, d) \mapsto \mathbb{R}$, such that the induced ranking maximizes the expected downstream utility. The reranker f_θ assigns a new relevance score $s_{q,d} = f_\theta(q, d)$ to each document $d \in D_q$.

3.2 Utility-driven Supervision

Following prior work ([38]), we define document utility by measuring the effect of including d in the LLM’s context on the downstream task-specific score (e.g., F1 or Exact Match). Formally, if y_q is the gold answer and $\hat{y}_q(d)$ is the model’s output when conditioned on d , then $U(q, d) = \mathcal{P}(y_q, \hat{y}_q(d))$, where $\mathcal{P}(\cdot)$ is a task-specific evaluation metric. The utilities $U(q, d)$ serve as supervision signals for training the reranker.

3.3 Feature Construction

Each query–document pair (q, d) is represented by a feature vector $\phi(q, d) \in \mathbb{R}^e$ such that: $\phi(q, d) = (f_1(q, d), f_2(q, d), \dots, f_e(q, d))$, where e is the total number of features. Specifically, we use lexical and IR-based features as follows.

Query statistics. Since longer queries may provide more discriminative context, we use the total number of query terms and the distinct number of query terms as features, i.e., $f_1 = |q|$ and $f_2 = |\{w \in q\}|$. We also use features that capture how “rare” or “specific” the query terms are, i.e., $f_3 = \min_{w \in q} IDF(w)$, $f_4 = \max_{w \in q} IDF(w)$ and $f_5 = \frac{1}{|q|} \sum_{w \in q} IDF(w)$.

Document statistics. We also use the corresponding features for documents, i.e., $f_6 = |d|$ and $f_7 = |\{w \in d\}|$. We use these features because longer documents might get unfairly favored because of their higher term frequencies. Similarly, we use the informativeness-related feature on the document side, i.e., $f_8 = \min_{w \in d} IDF(w)$, $f_9 = \max_{w \in d} IDF(w)$ and $f_{10} = \frac{1}{|d|} \sum_{w \in d} IDF(w)$.

Query–Document features. As a measure of lexical match, we use the number of overlapping terms as a feature, i.e., $f_{11} = q \cap d$. Furthermore, as a typically strong baseline for query–document matching, we use the retrieval score [32], i.e., if BM25 is used for retrieval, $f_{12} = BM25(q, d)$.

Topic features. Latent Dirichlet Allocation (LDA) [3] provides a lower-dimensional topical representation that captures co-occurrence patterns beyond surface word overlap. These topic-level signals help identify cases where a query and a candidate document are *semantically* aligned even if they share few exact terms, making them a complementary feature to lexical statistics. The ϕ vector for a specific topic is the probability distribution over all words in the vocabulary and the θ vector for a specific document is the probability distribution over all topics.

Let $\theta_q \in \mathbb{R}^M$ and $\theta_d \in \mathbb{R}^M$ be the topic distributions for query q and document d over M topics. We use cosine similarity to capture the semantic similarity

beyond words overlap, i.e., $f_{13} = \theta_q \cdot \theta_d / |\theta_q||\theta_d|$. Furthermore, in order to capture the intuition that if the query’s dominant topics are also well covered in the document, then the document is likely useful, we use the sum of document topic probabilities corresponding to the query’s top topics as a feature. Mathematically, $f_{14} = \sum_{m \in \text{Top}_a(\theta_q)} \theta_d(m)$, where $\text{Top}_a(\theta_q)$ denotes the top a topics in the query and $\theta_d(m)$ denotes topic m ’s probability in the document d .

3.4 Listwise Learning to Rank with LambdaMART

LambdaMART [7] is a listwise learning-to-rank algorithm that combines LambdaRank’s [6] gradient formulation with gradient-boosted regression trees (MART) [10]. For each query q , we have a candidate document set D_q with corresponding utility labels $\{U(q, d) | d \in D_q\}$. LambdaMART trains an ensemble of regression trees to predict scores $s_{q,d}$ such that the induced predicted ranking aligns with the ground-truth ranking based on the utilities. The objective is to minimize a loss that directly correlates with ranking quality. In LambdaMART, if d_i is ranked higher than d_j for the query q , the gradients are defined as follows:

$$\lambda_{ij} = -\sigma \cdot \frac{1}{1 + \exp^{\sigma(s_{q,d_i} - s_{q,d_j})}} \cdot \Delta NDCG_{ij}, \quad (1)$$

where, s_{q,d_i} and s_{q,d_j} are predicted scores for documents d_i and d_j respectively, $\sigma > 0$ is a scaling parameter and $\Delta NDCG_{ij}$ is the change in NDCG if d_i and d_j were swapped. The reranker updates its parameters using these λ values, which approximate the gradient of NDCG with respect to the predicted scores.

3.5 Inference

At test time, given a query, the first-stage retriever first retrieves N documents/passages. The trained lambdaMART model is then used to rank them and select top k . These k documents are then used to construct the prompt for the LLM. We note that while at training time we score each training example independently, at test time the LLM observes a prompt, i.e., a sequence of examples. We leave modeling the dependence between different training examples to future work.

4 Experiment Setup

4.1 Research Questions and Datasets

We investigate the following research questions:

- **RQ-1:** Does utility-driven reranking lead to a better downstream performance than traditional relevance-based retrieval across different LLMs and datasets?
- **RQ-2:** Does training a reranker with listwise ranking loss derived from LLM utility lead to a better downstream performance?

Table 1: Statistics of the datasets used in our experiments.

Dataset	Domain	#Train	#Dev	#Test
NQ-Open	Google queries	72,209	8,757	2,889
TriviaQA	Trivia questions	78,785	8,837	11,313

- **RQ-3:** Can a lightweight LambdaMART-based reranker (LURE-RAG) achieve competitive performance compared to dense neural rerankers (RePlug), while being more computationally efficient?
- **RQ-4:** Is the reranker transferrable, i.e., can the reranker trained on one LLM’s utilities perform well when used with other downstream LLMs of different sizes?

Datasets. We conduct our experiments on two standard open-domain question answering (QA) datasets, namely Natural Questions-Open (NQ-Open) and TriviaQA (see Table 1). More details on each dataset are as follows.

NQ-Open (NQ): The NQ-open dataset [27], a subset of the NQ dataset [26], differs by removing the restriction of linking answers to specific Wikipedia passages, thereby mimicking a more general information retrieval scenario similar to web searches. Following the methodology of [27], our primary source for answering queries is the English Wikipedia dump as of 20 December 2018. Consistent with the Dense Passage Retrieval (DPR) approach [25], each Wikipedia article in this dump is segmented into non-overlapping passages of 100 words.

TriviaQA (TQA): The TriviaQA dataset [23] is a large-scale benchmark consisting of question-answer pairs originally collected from trivia enthusiasts and quiz-league websites. Each question is accompanied by one or more evidence documents, either from Wikipedia or from web search results. For open-domain QA experiments, prior work [25,28,20] commonly discards the original evidence and evaluates on a standardized English Wikipedia corpus similar to NQ-Open. The “filtered” version ensures that all retained questions have at least one answer string present in the passage collection, yielding a cleaner evaluation set. Similar to [15,19], we perform evaluation on the dev set.

4.2 Methods Investigated

Baselines. We compare LURE-RAG¹ against the following baselines:

- **0-shot:** We use the base models as the baseline, i.e., without providing any retrieved context in the prompt.
- **k-shot:** We use the top- k documents/passages retrieved by the first-stage retriever. This baseline helps to check whether the notion of relevance in IR translates to RAG directly.
- **RePlug** [38]: The original RePlug framework optimizes a retriever by aligning retrieval scores with downstream LLM utility. However, RePlug has two

¹ Code available at <https://github.com/ManishChandra12/LURE-RAG>

practical limitations in our setting. First, it is incompatible with fixed or non-learnable retrievers such as BM25, since its learning relies on updating the retriever parameters. Second, it is resource-intensive, as the retriever optimization requires periodic index refreshes to incorporate updated document representations. To enable a fair comparison, we adapt RePlug into a black-box retriever setting: the retriever is fixed, and only a reranker is trained. Specifically, we retain RePlug’s KL-divergence-based objective, but apply it to align the reranker scores with downstream utility signals derived from the LLM outputs. This modified baseline preserves RePlug’s utility-driven supervision while removing its dependency on the retriever updates and index refreshes, making it directly comparable to our proposed LURE-RAG method.

Variants of LURE-RAG. To evaluate the effect of ranking loss in isolation (i.e., without efficiency consideration), we employ the following variants of LURE-RAG.

- **LURE-RAG:** We employ Phi-3-mini-4K-instruct [30] as the LLM for obtaining utilities in order to train the LambdaMART reranker.
- **UR-RAG:** We also introduce a dense variant, UR-RAG, where we replace LambdaMART with SBERT [36]. SBERT has been widely adopted for semantic retrieval and reranking, and it has been extensively benchmarked [39,16] in retrieval settings. We fine-tune SBERT using the following loss function [6,29] to inject the ranking signal into the reranker.

$$\mathcal{L}_{rank}(q) = \sum_{d_i, d_j \in D_q: U(q, d_i) > U(q, d_j)} \left(\frac{1}{r(d_i)} - \frac{1}{r(d_j)} \right) \log(1 + e^{(s_{q, d_j} - s_{q, d_i})}), \quad (2)$$

where $r(\cdot)$ is the ranking induced by $U(\cdot)$. If $U(q, d_i) > U(q, d_j)$, d_i is ranked higher than d_j and therefore, $r(d_i) < r(d_j)$. When d_i has a much higher rank than d_j , e.g., $r(d_i) = 1$ and $r(d_j) = 10$, the weight component of the loss will be high and the loss function will strongly draw $s(q, d_i)$ up from $s(q, d_j)$. This variant provides insights into the effectiveness-efficiency trade-off.

4.3 Evaluation Metrics and Hyper-parameter Configurations

Accuracy. Datasets like NQ-Open and TriviaQA provide a range of potential answers for each query. Frequently, these answers are different variants of the same concept (e.g., “President D. Roosevelt” or “President Roosevelt”). To evaluate the LLM accuracy on these datasets, which accept a range of valid answers for each query, we employ a binary assessment consistent with [24,9]. A response is deemed accurate if it contains at least one of the predefined correct answers; otherwise, it is inaccurate.

F1. F1-score is useful when answers involve partial overlap (e.g. multi-token answers), or when one wants to reward partial correctness or penalize missing parts of the answer. For a query q , precision and recall are defined as: $P_q = |y_q \cap \hat{y}_q| / |\hat{y}_q|$ and $R_q = |y_q \cap \hat{y}_q| / |y_q|$ respectively. We then define the F1 score

Table 2: A comparison between our proposed LURE-RAG approach and the baselines (using BM25 for retrieval) on the two datasets in terms of Accuracy (Acc) and F1-score. The best Acc and F1 values are bold-faced for each dataset and LLM combinations.

Dataset	Rank-By	Method	llama-1B		phi-3.8B		Qwen-14B	
			Acc	F1	Acc	F1	Acc	F1
NQ	Relevance	0-shot	.1215	.0616	.1970	.1026	.2323	.2518
		k-shot	.2876	.1590	.2676	.1866	.3548	.3498
	Utility	RePlug	.2932	.1894	.2895	.2007	.3775	.3700
		LURE-RAG	.2891	.1868	.2822	.1963	.3712	.3655
		LURE-RAG (w/o topics)	.2863	.1829	.2810	.1961	.3700	.3627
TQA	UR-RAG			.2999	.2068	.2933	.2088	.3822
	Relevance	0-shot	.2258	.2137	.2779	.2214	.3566	.3587
		k-shot	.2841	.2600	.3486	.3177	.3976	.3978
	Utility	RePlug	.3027	.3017	.3603	.3321	.4112	.4125
		LURE-RAG	.2968	.3010	.3569	.3272	.4057	.4047
		LURE-RAG (w/o topics)	.2947	.2973	.3544	.3261	.4008	.4001
	UR-RAG			.3111	.3104	.3676	.3472	.4186
				.4239				

as: $F1 = (1/|Q|) \sum_{q \in Q} \max_{y_q \in C_q} (2P_q R_q / (P_q + R_q))$, where C_q denotes the set of all available ground-truth answers for the query q .

LURE-RAG relies on the LLM predictions for the reranker training. To analyse the variations that may be caused due to this choice of LLM, we conduct our experiments on three different LLMs - each from different families. The objective is to analyse the variations in LURE-RAG's performance corresponding to different sizes of the models, and variations in characteristics of models across different families, thereby allowing to answer RQ-4. In particular, we use Meta's Llama-3.2-1B-Instruct [33], Microsoft's Phi-3-mini-3.8B-4K-instruct [1] and Alibaba's Qwen2.5-14B-Instruct-1M [35]. Furthermore, we conduct two sets of experiments - one involving BM25 [37] for sparse retrieval and the other involving Contriever [21] for dense retrieval.

In our experiments, we set the number of first-stage retrieved documents using BM25/Contriever (N) to 10 and RAG context size (k) to 5. As a reranker for RePlug baseline and UR-RAG, we finetune SBERT all-MiniLM-L6-v2 [36]. For fine-tuning SBERT, we use a training batch size of 32, a learning rate of 1e-5 and early stopping with a patience of 3 epochs. As an optimiser, we use Adam with weight decay. The optimal values of the hyperparameters are obtained via a grid search. For LDA, we use a random subset of 1M documents from the corpus and set the number of topics M to 100 and the threshold for 'top' topics (a) to 20.

Table 3: A comparison between our proposed LURE-RAG approach and the baselines (using Contriever for retrieval) on the two datasets in terms of Accuracy (Acc) and F1-score. The best Acc and F1 values are bold-faced for each dataset and LLM combinations.

Dataset	Rank-By	Method	llama-1B		phi-3.8B		Qwen-14B	
			Acc	F1	Acc	F1	Acc	F1
NQ	Relevance	0-shot	.1215	.0616	.1970	.1026	.2323	.2518
		k-shot	.2911	.1782	.2631	.1835	.3508	.3401
	Utility	RePlug	.2976	.1853	.2799	.1994	.3751	.3696
		LURE-RAG	.2945	.1811	.2722	.1949	.3687	.3669
		LURE-RAG (w/o topics)	.2940	.1798	.2720	.1904	.3666	.3664
	UR-RAG		.3027 .1915 .2883 .2076 .3811 .3800					
TQA	Relevance	0-shot	.2258	.2137	.2779	.2214	.3566	.3587
		k-shot	.2834	.2620	.3509	.3137	.3966	.3954
	Utility	RePlug	.3111	.3042	.3582	.3400	.4010	.4077
		LURE-RAG	.3059	.2951	.3635	.3304	.3994	.4039
		LURE-RAG (w/o topics)	.2994	.2897	.3628	.3220	.4010	.4017
	UR-RAG		.3184 .3098 .3699 .3497 .4177 .4203					

5 Results

Tables 2 and 3 report the performance of our proposed method relative to the baselines on two open-domain QA benchmarks, Natural Questions (NQ) and TriviaQA (TQA), across three different LLMs of increasing model sizes — llama-1B, phi-3.8B, and Qwen-14B.

Comparison between relevance-based and utility-driven paradigms. Across the dataset and LLM choices, we observe that utility-driven approaches consistently outperform the relevance-based ones. While k-shot prompting improves over 0-shot in the relevance-based setting (e.g., +0.16 Acc improvement for llama-1B on NQ using sparse retrieval), the gap between k-shot and utility-driven reranking remains substantial. This confirms that document ordering based solely on classical IR relevance is insufficient for optimal downstream performance, and utility-driven reranking better aligns retrieval with the LLM’s actual answer quality. Therefore, in answer to **RQ-1**, we conclude that utility-driven reranking leads to better downstream performance than traditional relevance-based retrieval across different LLMs and datasets.

Effect of listwise ranking loss. While RePlug remains a strong utility-driven baseline, we observe that UR-RAG significantly outperforms it across all settings. For example, on NQ with llama-1B using sparse retrieval, RePlug achieves 0.1894 F1, whereas UR-RAG achieves 0.2068, yielding noticeable improvements. Paired t-tests at 5% significance level reveal that the improvements in the evaluation

metrics obtained using UR-RAG are statistically significant across all dataset and LLM combinations (tests are conducted separately) compared to RePlug. A key reason for this gain lies in the ranking loss employed by UR-RAG, which explicitly encourages the reranker to preserve correct document orderings according to the downstream utility. In contrast, RePlug does not incorporate any ranking-aware objective, relying solely on the KL divergence between the reranker scores and utility distributions. This difference in performance is observed because order of documents matter in RAG [9]. Therefore, in answer to **RQ-2**, we conclude that training a reranker with listwise ranking loss derived from LLM utility leads to better downstream performance.

Comparison between dense and light-weight reranker. LURE-RAG achieves results that are competitive with the dense reranker baseline (RePlug), while requiring only a lightweight LambdaMART model for training and inference. The F1-scores and accuracies for the NQ dataset using LURE-RAG (in sparse retrieval setup) are always above 97.8% and 97.4%, respectively, of that achieved using RePlug. Similar numbers for TQA dataset are 98.1% and 98.05%, respectively. These results indicate that a lightweight reranker, when guided by *list-wise ranking* utility-driven supervision provides a much more efficient alternative with only minor effectiveness trade-offs compared to a dense neural reranker not trained with a ranking loss. Therefore, in answer to **RQ-3**, we conclude that a lightweight LambdaMART-based reranker can achieve competitive performance compared to dense neural rerankers that are trained without ranking loss, while being more computationally efficient.

Along the expected lines, UR-RAG, which fine-tunes SBERT with a utility-driven ranking loss, yields the highest accuracy and F1 scores across all dataset-model combinations. For instance, on TQA with Qwen-14B (in sparse retrieval setup), UR-RAG achieves 0.4186 accuracy and 0.4239 F1, the strongest results in Table 2. The F1-scores and accuracies for the NQ dataset using LURE-RAG are always above 90.3% and 96.2%, respectively, of that achieved using UR-RAG. Similar numbers for TQA dataset are 94.2% and 95.4%, respectively. These results indicate that when a lightweight reranker and a dense neural reranker, both trained using a ranking loss, are compared, the performance deterioration is larger.

Furthermore, we also see improvements across different LLMs. Note that we train the reranker using the utilities obtained from the phi-3 model, and reuse the same trained reranker for other LLMs as well. Therefore, in answer to **RQ-4**, we conclude that the reranker trained on one LLM’s utilities can perform well when used with other downstream LLMs of different sizes.

Interpretability. The benefit of using a lightweight model like lambdaMART is that it’s interpretable. To understand which signals drive the reranker’s decisions, we analyse the feature importance scores from the trained LambdaMART models (see Figure 2). Feature importances are computed using the gain metric, which measures the relative contribution of each feature to the model’s decision splits. For every split in the trees, the “gain” (improvement in the objective function)

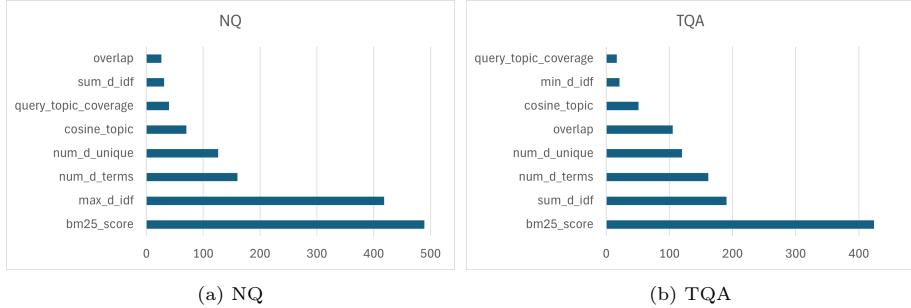


Fig. 2: Feature importance scores of the top-8 features obtained from the trained lambdamART model for LURE-RAG.

is calculated. The total gain for a feature is summed across all splits where it is used, providing a measure of its importance. The most influential feature is the BM25 similarity score (`bm25_score`), which dominates the learned model. This suggests that classical lexical similarity remains a strong signal for utility-driven reranking. Beyond this, the number of (unique) document terms and document-level IDF statistics also exhibit notable importance, indicating that documents containing rare, high-value terms often align better with downstream LLM utility. Moreover, query-document term overlap and features derived from LDA provide additional, though relatively weaker, signals.

Ablation study. To examine the contribution of the LDA topic features, we conduct an ablation study by removing them from the feature set. As expected from the feature importance analysis (Figure 2), topic features exhibit relatively low importance compared to lexical and BM25-based features. Consistently, we observe only marginal degradation in performance across LLMs when these features are excluded, as shown in Tables 2 and 3 corresponding to ‘LURE-RAG (w/o topics)’ rows. This result suggests that while topic-level information can provide additional semantic signal, the reranker primarily relies on stronger lexical and retrieval-based signals. Nonetheless, the inclusion of LDA features does offer small but consistent improvements.

6 Conclusions

In this work, we proposed LURE-RAG, a lightweight utility-driven reranking framework that improves retrieval-augmented generation by incorporating a ranking-aware objective, offering a more efficient alternative to existing approaches. We further introduced UR-RAG, a dense variant that achieves the strongest results, significantly improving accuracy and F1 by up to 3%. Our results demonstrate that utility-driven reranking using principled ranking losses provides a powerful and practical approach to aligning retrieval with downstream generation tasks in retrieval-augmented generation. In future work, we aim to explore richer feature representations including semantic signals.

Disclosure of Interests. The authors have no competing interests to declare that are relevant to the content of this article.

References

1. Abdin, M.I., Ade Jacobs, S., Awan, A.A., Aneja, J., Awadallah, A., Hassan Awadalla, H., Bach, N., Bahree, A., Bakhtiari, A., Behl, H., Benhaim, A., Bilenko, M., Bjorck, J., Bubeck, S., Cai, M., Mendes, C.C.T., Chen, W., Chaudhary, V., Chopra, P., Giorno, A.D., de Rosa, G., Dixon, M., Eldan, R., Iter, D., Goswami, A., Gunasekar, S., Haider, E., Hao, J., Hewett, R.J., Huynh, J., Javaheripi, M., Jin, X., Kauffmann, P., Karampatziakis, N., Kim, D., Khademi, M., Kurilenko, L., Lee, J.R., Lee, Y.T., Li, Y., Liang, C., Liu, W., Lin, X.E., Lin, Z., Madan, P., Mitra, A., Modi, H., Nguyen, A., Norick, B., Patra, B., Perez-Becker, D., Portet, T., Pryzant, R., Qin, H., Radmilac, M., Rosset, C., Roy, S., Saarikivi, O., Saied, A., Salim, A., Santacroce, M., Shah, S., Shang, N., Sharma, H., Song, X., Ruwase, O., Wang, X., Ward, R., Wang, G., Witte, P., Wyatt, M., Xu, C., Xu, J., Xu, W., Yadav, S., Yang, F., Yang, Z., Yu, D., Zhang, C., Zhang, C., Zhang, J., Zhang, L.L., Zhang, Y., Zhang, Y., Zhou, X.: Phi-3 technical report: A highly capable language model locally on your phone. Tech. Rep. MSR-TR-2024-12, Microsoft (August 2024), <https://www.microsoft.com/en-us/research/publication/phi-3-technical-report-a-highly-capable-language-model-locally-on-your-phone/>
2. Ai, Q., Bi, K., Guo, J., Croft, W.B.: Learning a deep listwise context model for ranking refinement. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. pp. 135–144 (2018)
3. Blei, D.M., Ng, A.Y., Jordan, M.I.: Latent dirichlet allocation. Journal of Machine Learning Research (JMLR) **3**, 993–1022 (2003). <https://doi.org/10.1162/jmlr.2003.3.4-5.993>
4. Brown, A., Roman, M., Devereux, B.: A systematic literature review of retrieval-augmented generation: Techniques, metrics, and challenges. Applied Sciences **9**(12) (2025). <https://doi.org/10.48550/arxiv.2508.06401>
5. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning. p. 89–96. ICML ’05, Association for Computing Machinery, New York, NY, USA (2005). <https://doi.org/10.1145/1102351.1102363>, <https://doi.org/10.1145/1102351.1102363>
6. Burges, C.J.C., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: Proceedings of the 20th International Conference on Neural Information Processing Systems. p. 193–200. NIPS’06, MIT Press, Cambridge, MA, USA (2006)
7. Burges, C.J.: From ranknet to lambdarank to lambdamart: An overview. Tech. Rep. MSR-TR-2010-82, Microsoft Research (2010), <https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/>
8. Chandra, M., Ganguly, D., Ounis, I.: One size doesn’t fit all: Predicting the number of examples for in-context learning. In: Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part I. p. 67–84. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-88708-6_5, https://doi.org/10.1007/978-3-031-88708-6_5

9. Cuconasu, F., Trappolini, G., Siciliano, F., Filice, S., Campagnano, C., Maarek, Y., Tonelotto, N., Silvestri, F.: The power of noise: Redefining retrieval for rag systems. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. p. 719–729. SIGIR ’24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3626772.3657834>
10. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. *Annals of Statistics* **29**(5), 1189–1232 (2001). <https://doi.org/10.1214/aos/1013203451>
11. Gao, J., Li, L., Ji, K., Li, W., Lian, Y., yuzhuo fu, Dai, B.: SmartRAG: Jointly learn RAG-related tasks from the environment feedback. In: The Thirteenth International Conference on Learning Representations (2025), <https://openreview.net/forum?id=0Cd3cffulp>
12. Gao, Y., Xiong, Y., Gao, X., Jia, K., Pan, J., Bi, Y., Dai, Y., Sun, J., Wang, M., Wang, H.: Retrieval-augmented generation for large language models: A survey (2024), <https://arxiv.org/abs/2312.10997>
13. Glass, M., Rossiello, G., Chowdhury, M.F.M., Naik, A., Cai, P., Gliozzo, A.: Re2G: Retrieve, rerank, generate. In: Carpuat, M., de Marneffe, M.C., Meza Ruiz, I.V. (eds.) Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. pp. 2701–2715. Association for Computational Linguistics, Seattle, United States (Jul 2022). <https://doi.org/10.18653/v1/2022.naacl-main.194>, <https://aclanthology.org/2022.naacl-main.194/>
14. Gupta, S., Ranjan, R., Singh, S.N.: A comprehensive survey of retrieval-augmented generation (rag): Evolution, current landscape and future directions (2024), <https://arxiv.org/abs/2410.12837>
15. Guu, K., Lee, K., Tung, Z., Pasupat, P., Chang, M.W.: Retrieval-augmented language model pre-training. In: International Conference on Machine Learning (ICML). pp. 3929–3938. PMLR (2020)
16. Hofstätter, S., Althammer, S., Khattab, O., Mitra, B., Hanbury, A.: Efficiently teaching an effective dense retriever with balanced topic aware sampling. In: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 113–122. ACM (2021)
17. Hu, Z., Wang, Y., Peng, Q., Li, H.: Unbiased lambdamart: An unbiased pairwise learning-to-rank algorithm. In: Proceedings of The World Wide Web Conference (WWW ’19). pp. 2271–2281 (2019). <https://doi.org/10.1145/3308558.3313768>
18. Huang, Y., Huang, J.: A survey on retrieval-augmented text generation for large language models (2024), <https://arxiv.org/abs/2404.10981>
19. Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., Grave, E.: Atlas: Few-shot learning with retrieval augmented language models. In: International Conference on Machine Learning (ICML). pp. 13182–13210. PMLR (2023)
20. Izacard, G., Grave, E.: Leveraging passage retrieval with generative models for open domain question answering. In: International Conference on Learning Representations (ICLR) (2021), <https://arxiv.org/abs/2007.01282>
21. Izacard, G., Lewis, P., Riedel, S., Karpukhin, V., Minervini, P., Petroni, F., Grave, E.: Unsupervised dense information retrieval with contrastive learning. *Transactions of the Association for Computational Linguistics* **10**, 665–681 (2022)
22. Jeong, S., Baek, J., Cho, S., Hwang, S.J., Park, J.: Adaptive-RAG: Learning to adapt retrieval-augmented large language models through question complex-

- ity. In: Duh, K., Gomez, H., Bethard, S. (eds.) Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers). pp. 7035–7049. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024). <https://doi.org/10.18653/v1/2024.naacl-long.389>, <https://aclanthology.org/2024.naacl-long.389>
23. Joshi, M., Choi, E., Weld, D., Zettlemoyer, L.: 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). pp. 1601–1611. Association for Computational Linguistics (2017). <https://doi.org/10.18653/v1/P17-1147>, <https://aclanthology.org/P17-1147>
 24. Kandpal, N., Deng, H., Roberts, A., Wallace, E., Raffel, C.: Large language models struggle to learn long-tail knowledge. In: Krause, A., Vreeken, J., Yoav, S. (eds.) Proceedings of the 40th International Conference on Machine Learning. vol. 202, pp. 15886–15904. PMLR (2023)
 25. Karpukhin, V., Oguz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., Yih, W.t.: Dense passage retrieval for open-domain question answering. In: Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP). pp. 6769–6781. Association for Computational Linguistics (2020). <https://doi.org/10.18653/v1/2020.emnlp-main.550>, <https://aclanthology.org/2020.emnlp-main.550>
 26. Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.W., Dai, A.M., Uszkoreit, J., Le, Q., Petrov, S.: Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics* **7**, 453–466 (2019)
 27. Lee, K., Chang, M.W., Toutanova, K.: Latent retrieval for weakly supervised open domain question answering. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics. pp. 6086–6096. Association for Computational Linguistics (2019). <https://doi.org/10.18653/v1/P19-1612>, <https://aclanthology.org/P19-1612>
 28. Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.t., Rocktäschel, T., Riedel, S., Kiela, D.: Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS ’20, Curran Associates Inc., Red Hook, NY, USA (2020)
 29. Li, X., Lv, K., Yan, H., Lin, T., Zhu, W., Ni, Y., Xie, G., Wang, X., Qiu, X.: Unified demonstration retriever for in-context learning. In: Rogers, A., Boyd-Graber, J., Okazaki, N. (eds.) Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 4644–4668. Association for Computational Linguistics, Toronto, Canada (Jul 2023), <https://aclanthology.org/2023.acl-long.256>
 30. Li, Y., Bubeck, S., Eldan, R., Del Giorno, A., Gunasekar, S., Lee, Y.T.: Textbooks are all you need ii: **phi-1.5** technical report. arXiv preprint arXiv:2309.05463 (2023)
 31. Liu, N.F., Lin, K., Hewitt, J., Paranjape, A., Bevilacqua, M., Petroni, F., Liang, P.: Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics* **12**, 157–173 (2024). https://doi.org/10.1162/tacl_a_00638, <https://aclanthology.org/2024.tacl-1.9/>

32. Liu, T.Y.: Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval* **3**(3), 225–331 (2009). <https://doi.org/10.1561/1500000016>
33. Meta AI: Llama 3.2: Revolutionizing edge ai and vision with open, customizable models. Tech. rep., Meta Platforms, Inc. (Sep 2024), <https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/>, accessed: 2025-05-22
34. Pasumarthi, R.K., Bruch, S., Wang, X., Bendersky, M., Najork, M., Wang, J., Li, Y., Anil, R., Cheng, H., Chappidi, R., et al.: Tf-ranking: Scalable tensorflow library for learning-to-rank. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 2970–2978 (2019)
35. Qwen Team: Qwen2.5 technical report. arXiv preprint arXiv:2412.15115 (2024), submitted December 2024
36. Reimers, N., Gurevych, I.: Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In: Inui, K., Jiang, J., Ng, V., Wan, X. (eds.) *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. pp. 3982–3992. Association for Computational Linguistics, Hong Kong, China (Nov 2019). <https://doi.org/10.18653/v1/D19-1410>, <https://aclanthology.org/D19-1410>
37. Robertson, S.E., Zaragoza, H.: The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval* **3**(4), 333–389 (2009). <https://doi.org/10.1561/1500000019>
38. Shi, W., Min, S., Yasunaga, M., Seo, M., James, R., Lewis, M., Zettlemoyer, L., Yih, W.t.: REPLUG: Retrieval-augmented black-box language models. In: Duh, K., Gomez, H., Bethard, S. (eds.) *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*. pp. 8371–8384. Association for Computational Linguistics, Mexico City, Mexico (Jun 2024). <https://doi.org/10.18653/v1/2024.naacl-long.463>, <https://aclanthology.org/2024.naacl-long.463>
39. Thakur, N., Reimers, N., Daxenberger, J., Gurevych, I.: BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In: *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. pp. 2308–2318. ACM (2021)
40. Tian, F., Ganguly, D., Macdonald, C.: Is relevance propagated from retriever to generator in rag? In: *Advances in Information Retrieval: 47th European Conference on Information Retrieval, ECIR 2025, Lucca, Italy, April 6–10, 2025, Proceedings, Part I*. p. 32–48. Springer-Verlag, Berlin, Heidelberg (2025). https://doi.org/10.1007/978-3-031-88708-6_3, https://doi.org/10.1007/978-3-031-88708-6_3
41. Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures. *Information Retrieval* **13**(3), 254–270 (2010). <https://doi.org/10.1007/s10791-009-9112-1>
42. Zhang, H., Zhang, R., Guo, J., de Rijke, M., Fan, Y., Cheng, X.: Are large language models good at utility judgments? In: *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. p. 1941–1951. SIGIR '24, Association for Computing Machinery, New York, NY, USA (2024). <https://doi.org/10.1145/3626772.3657784>, <https://doi.org/10.1145/3626772.3657784>