LTRR: Learning To Rank Retrievers for LLMs

To Eun Kim
Carnegie Mellon University
Pittsburgh, PA, USA
toeunk@cs.cmu.edu

Fernando Diaz
Carnegie Mellon University
Pittsburgh, PA, USA
diazf@acm.org

Abstract

Retrieval-Augmented Generation (RAG) systems typically rely on a single fixed retriever, despite growing evidence that no single retriever performs optimally across all query types. In this paper, we explore a query routing approach that dynamically selects from a pool of retrievers based on the query, using both train-free heuristics and learned routing models. We frame routing as a learning-to-rank (LTR) problem and introduce LTRR, a framework that learns to rank retrievers by their expected utility gain to downstream LLM performance. Our experiments, conducted on synthetic QA data with controlled query type variations, show that routing-based RAG systems can outperform the best single-retriever-based systems. Performance gains are especially pronounced in models trained with the Answer Correctness (AC) metric and with pairwise learning approaches, especially with XGBoost. We also observe improvements in generalization to out-of-distribution queries. As part of the SIGIR 2025 LiveRAG challenge, our submitted system demonstrated the practical viability of our approach, achieving competitive performance in both answer correctness and faithfulness. These findings highlight the importance of both training methodology and metric selection in query routing for RAG systems.

Keywords

Retrieval-Augmented Generation, Learning to Rank, Query Routing

1 Introduction

Retrieval-Augmented Generation (RAG) [28, 30] enhances Large Language Models (LLMs) by leveraging external knowledge. However, despite the availability of numerous retrieval models, no single retriever consistently outperforms others across diverse queries and corpora [25, 40]. Inspired by distributed information retrieval and federated search, the research community has recently begun investigating *query routing* as a means to address this challenge.

Effective query routing is essential given the complexity introduced by various retrieval methods. Optimal routing can improve adaptability across query types, thereby enhancing downstream RAG performance. Additionally, selective routing can help reduce computational costs by activating only necessary retrievers, including potentially bypassing retrieval entirely. Furthermore, modern retrieval systems increasingly operate as independent, competing services, effectively forming a *search marketplace* targeted at machine consumers (LLMs) [31], making efficient query routing a timely research.

Existing query routing approaches exhibit several limitations. Many rely on heuristics [25, 29] or optimize solely for traditional retrieval metrics intended for human end-users [20], while in the context of LLM consumers, routing strategies should ideally optimize downstream LLM performance rather than traditional retrieval metrics [27, 37]. Additionally, these methods often leverage query-corpus similarity-based routing [20, 25, 26, 29], which becomes impractical in *uncooperative* environments [2], where we do not have access to the corpus, and the retrievers are assumed to only provide a search interface (e.g., Model Context Protocol [1]).

In this paper, we propose **Learning to Rank Retrievers (LTRR)**, a novel query routing approach explicitly optimized for downstream LLM utility. Unlike previous work, our ranking of retrievers is based directly on the relative improvement in the LLM's generation quality compared to a no-retrieval baseline. This approach inherently addresses both retriever-selection ("where to query") and retrieval-necessity ("when to query"), as no-retrieval is explicitly included as one of the routing options. We comprehensively evaluate multiple learning-to-rank (LTR) algorithms using diverse query sets, focusing on retrievers differentiated by retrieval strategies rather than corpus content. This setting reflects contemporary retrieval environments, where systems typically utilize similar large-scale corpora but differ significantly in their retrieval approaches.

Our experiments demonstrate that LTRR algorithms, particularly those trained using pairwise XGBoost, significantly outperform the best standard single-retriever-based RAG systems in both in-distribution and some out-of-distribution evaluations. We present our system and findings at the SIGIR 2025 LiveRAG Workshop, where our work was selected for a spotlight talk, and release our code to encourage further research in this direction.[1]

2 Related Work

Distributed IR. Our approach to query routing builds on several established IR fields, including Distributed Search, Federated Search, Selective Search, Aggregated Search, and Meta Search. Distributed search traditionally address the selection of relevant document collections based on query relevance, often in a distributed and disjoint environment [4, 5]. Federated and selective search extend these ideas, focusing on brokered retrieval across multiple independent and typically uncooperative systems, employing resource representations and selection strategies to effectively route queries [11, 15]. Aggregated Search similarly aims to integrate diverse retrieval results from specialized vertical search services into a unified search interface, emphasizing the selection of relevant services per query [2]. Additionally, Meta Search combines results from several search engines to improve overall relevance, recognizing that no single search engine consistently outperforms others across diverse queries [7, 19].

While our query routing methodology shares conceptual similarities with these fields, it uniquely differs in its explicit emphasis on routing queries to varied retrieval strategies optimized directly for downstream retrieval-augmented generation (RAG) tasks.

[1] https://github.com/kimdanny/Starlight-LiveRAG

Query Routing Strategies. Building on insights from distributed IR, recent RAG systems increasingly incorporate query routing strategies. Khramtsova et al. [25] examined various dense retriever selection methods and identified corpus similarity as a reliable, training-free criterion. This line of work was extended by Khramtsova et al. [26], who proposed an unsupervised approach using pseudo-query generation and LLM-based pseudo relevance judgments to rank dense retrievers.

Mu et al. [32] proposed a routing strategy that directly predicts downstream LLM performance, bypassing traditional retrieval effectiveness metrics. However, this approach overlooks cases where retrieval may not be beneficial and struggles with the variability of absolute score prediction across queries. Similarly, Adaptive-RAG [22] classifies queries by their perceived complexity to select retrieval strategies, but this relies on human-centric definitions of complexity and requires curated training data, which may not align with LLM behavior. Other recent studies expand the space of query routing. RouterRetriever [29] uses embedding similarities for retriever selection. Guerraoui et al. [20] introduced RAGRoute, a lightweight classifier that dynamically selects retrieval sources to optimize recall and classification accuracy. Tang et al. [38] framed routing as a contextual multi-armed bandit problem in knowledge graph-based RAG systems, but without modeling no-retrieval as a viable option.

Our approach emphasizes learning to rank retrievers based directly on improvements in downstream LLM utility. It explicitly includes no-retrieval as a valid action and is evaluated over a diverse set of retrieval strategies.

3 RAG Method

Multi-Retriever Setup. We utilize Opensearch sparse BM25 and Pinecone dense E5$_{base}$ retrievers as a base retrievers, and we combine two reranking strategies with distinct goals to make variations in retrieval strategies. The first, *score regularization* [14], focuses on improving retrieval performance. The second, *stochastic reranking* [27], aims to enhance item-fairness and diversity, which can also improve downstream RAG performance. As a result, we establish six distinct retrievers: (1) BM25; (2) BM25 + Score Regularization Reranking; (3) BM25 + Stochastic Reranking; (4) E5$_{base}$; (5) E5$_{base}$ + Score Regularization Reranking; and (6) E5$_{base}$ + Stochastic Reranking. Details of reranking methods can be found in Appendix A.

All retrievers utilize the sampled version of FineWeb corpus (15M documents)[35], and their retrieval strategies and corpus statistics remain hidden from both the generator and router (uncooperative environment).

Query Routing via LTRR. Our RAG framework routes queries to a suitable retriever $\mathcal{R}_i$ from a pool of multiple retrievers $L_{\mathcal{R}}$. For each input instance x, we generate a query q via a query generation function $\phi_q(x)$. The core objective is to route this query to one or more retrievers that maximize the downstream performance of the RAG system. Formally, we introduce a router function $\mathcal{F}$ that maps queries to a ranked set of retrievers:

$$\mathcal{F}(q; L_{\mathcal{R}}) \rightarrow \pi_{L_{\mathcal{R}}}, \tag{1}$$

where $\pi_{L_{\mathcal{R}}}$ is a ranking of retrievers reflecting the predicted utility of each retriever can give to the downstream generator $\mathcal{G}$ for the given query. In our implementation, we route queries to the top-ranked retriever using a pairwise XGBoost-based router (chosen for its empirical effectiveness, as discussed in later sections). Importantly, we include a 'no-retrieval' option $\mathcal{R}_0$ in the ranking. This allows the system to bypass retrieval altogether when the router predicts that relying solely on the language model's parametric memory yields the best performance.

Generator. We use Falcon3-10B-Instruct [39] as the generator in our RAG system. Inspired by recent work on prompting LLMs to reason with external information [8, 23], we design prompts that instruct the model to explicitly assess the relevance and utility of each retrieved passage. Specifically, the model is prompted to reflect on how to use the passages in a `<think>` section, followed by its final answer in a `<answer>` section. We extract only the content within the `<answer>` tag as the system's output. For ill-formatted generations, a fallback prompt omitting explicit reasoning is used. Full prompt details are provided in Appendix B.

4 Learning to Rank Retrievers

We propose Learning to Rank Retrievers (LTRR4LLM), which formulates the routing problem as a learning-to-rank task tailored specifically to optimize downstream LLM performance.

To first derive the ground-truth retriever rankings required for training, we measure the utility gain δ_i achieved by retriever $\mathcal{R}_i$ relative to the baseline generator performance (without retrieval):

$$\delta_i = \mu_u(\mathcal{G}(\overline{x}_i), y) - \mu_u(\mathcal{G}(x), y), \tag{2}$$

where $\overline{x}_i = \phi_p(x, \mathcal{R}_i(q, k))$ with k denoting the number of passages to retrieve, ϕ_p denoting a prompt construction function for an LLM $\mathcal{G}$, and μ_u is an arbitrary string utility metric. To ensure comparability across queries, utility-gain scores are min-max normalized per query into the range of [0,1].

Following the LTR literature [6], LTRR is then characterized by a scoring function f:

$$f(\Phi(q, \mathcal{R}_i)) \rightarrow \mathbb{R} \tag{3}$$

that assigns a score to each retriever based on query- and retriever-specific features $\Phi(q, \mathcal{R}_i)$ extracted from the i'th retriever $\mathcal{R}_i$.

To train the ranking model, we experiment with three well-established approaches (detailed in Appendix C): (1) the *pointwise* approach, where the model predicts each retriever's utility gain δ_i independently using a regression loss; (2) the *pairwise* approach, where the model learns to minimize ranking inversions between retriever pairs based on their relative utility gains compared to the no-retrieval baseline; and (3) the *listwise* approach, which directly optimizes the predicted utility gains over the full set of retrievers for each query.

4.1 LTR Features

Our setup assumes an *uncooperative* retrieval environment in which retrievers do not expose detailed corpus statistics or embedding model specifications. Thus, we extract a set of query-dependent pre-retrieval features and query- and retriever-dependent post-retrieval features to facilitate effective learning-to-rank (LTR) modeling.

For **pre-retrieval features**, we include the query representation ($\mathbb{R}^{\text{dim}}$), query length, and query type. The query representation is a vector produced by an embedding model, with optional dimensionality reduction (e.g., via PCA).[2] Query type is determined using a lightweight classifier that distinguishes between keyword-based and natural language queries.[3] These features are query-specific but not retriever-specific, allowing LTRR models to learn differences across queries.

Post-retrieval features, in contrast, are computed after querying all retrievers and are both query- and retriever-specific, providing the LTRR model with signals to differentiate between retrievers.

Let $z_i = [d_{i,1}, \ldots, d_{i,k}]$ denote the top-k documents retrieved by retriever $\mathcal{R}_i$, $s(\cdot,\cdot)$ be an embedding-based cosine similarity function, and M be the total number of available retrievers. We define $e(z_i) = \frac{1}{k}\sum_{j=1}^{k} \texttt{embed}(d_{i,j})$ as the aggregated semantic embedding of the retrieved documents retrieved by $\mathcal{R}_i$. Using these definitions, we construct the following semantic and statistical features:

- `OverallSim`: similarity between query and the aggregated embedding of retrieved documents, $s(q, e(z_i))$,
- `AvgSim`: average similarity score between query and individual retrieved documents, $\texttt{avg}_j(s(q, d_{i,j}))$,
- `MaxSim`: maximum similarity score between query and individual retrieved documents, $\texttt{max}_j(s(q, d_{i,j}))$,
- `VarSim`: variance of retrieval similarity scores, $\texttt{var}_j(s(q, d_{i,j}))$, capturing retrieval confidence dispersion,
- `Moran`: Moran coefficient [13], which measures semantic autocorrelation among retrieved documents in alignment with the cluster hypothesis, and
- `CrossRetSim`: average semantic similarity of the current retriever's result set with those from other retrievers, defined as $\frac{1}{M-1}\sum_{m;m\neq i}^{M} s(e(z_i), e(z_m))$, which can indicate a uniqueness of a ranking compared to other rankings from other retrievers.

For the no-retrieval option ($\mathcal{R}_0$), only pre-retrieval features are available. To maintain consistent feature dimensionality across retrievers, we handle missing post-retrieval features differently depending on the model type. For neural LTR models, we introduce a dedicated learnable parameter vector to represent the post-retrieval feature space of $\mathcal{R}_0$. This vector is randomly initialized and optimized during training, allowing the model to implicitly encode the characteristics of the no-retrieval strategy. For non-neural models, we apply median imputation based on the training data to fill in the missing post-retrieval features, ensuring compatibility with fixed-length feature inputs.

5 Experiment

We evaluate our proposed routing approaches against a range of baseline and train-free routing models.

5.1 Routing Models

We first consider five heuristic, train-free routing models, each based on a post-retrieval feature: `OverallSim`, `AvgSim`, `MaxSim`, `VarSim` (where lower variance is preferred), and `Moran`.

For learned routing models trained via the LTRR framework, we evaluate eleven models spanning the **pointwise**, **pairwise**, and **listwise** paradigms. In the pointwise and pairwise settings, we train models using XGBoost [9], SVM^{rank} [24], a feedforward network (FFN), and DeBERTa [21]. In the listwise setting, we evaluate ListNet [6], LambdaMart [42], and DeBERTa-based models.

All LTRR models are trained using utility labels derived from two metrics: BEM [3] and Answer Correctness (AC) [17], both shown to correlate strongly with human evaluations in RAG setting [34].

5.2 Datasets

We generate a synthetic dataset using DataMorgana [18], enabling fine-grained control over question and user characteristics. Question configurations include four dimensions: answer type, premise, phrasing, and linguistic variation. User configurations are based on expertise level (expert vs. novice). Full dataset generation details are provided in Appendix D.

For our LTRR experiments, we focus on the answer-type category, which comprises five distinct question types: factoid, multi-aspect, comparison, complex, and open-ended.[4]

We construct five dataset splits for evaluation. The **Balanced split** includes all question types proportionally in both training and test sets. Four **unseen type splits** (multi-aspect, comparison, complex, and open-ended) each hold out one question type from training and use it exclusively for testing, enabling us to assess model generalization to unseen query types. Dataset statistics are reported in Appendix E.

6 Results and Discussion

RQ1: Do routing-based RAG systems outperform the best-performing standard RAG model? To study this question, we first identify the best-performing single-retriever (standard) RAG system for each dataset under the two utility metrics. Their downstream scores are shown in the 'Best Standard RAG' row of Table 1 (see Appendix F for details).

As shown in Table 1, the train-free routing models did not yield statistically significant improvements over the best-performing standard RAG systems, despite showing some numerical gains. In contrast, the LTRR-based models demonstrated more substantial improvements, particularly with the pointwise SVM^{rank} and DeBERTa, as well as the pairwise XGBoost and DeBERTa models, which achieved statistically significant gains on the Balanced split.

However, performance gains were noticeably higher for router models trained using the AC utility metric compared to those trained with BEM. Statistically significant improvements were observed only for AC-based routers (highlighted in bold), while no such gains were found for BEM-based models. We attribute this discrepancy to differences in metric reliability: although both BEM and AC correlate well with human judgments, prior work shows that AC consistently achieves stronger alignment [34]. Since LTRR models are trained directly on utility labels, the choice of a consistent and accurate metric is critical.

RQ2: Do LTRR-based routing algorithms outperform the best-performing train-free routing model? We also examined

[2] We use the E5$_{base}$ model to extract query embeddings, which are then reduced to 32 dimensions using PCA. For DeBERTa-based models, however, we retain the original, non-reduced embeddings.

[3] https://huggingface.co/shahrukhx01/bert-mini-finetune-question-detection

[4] Complex question type is configured in user configuration under expert user category.

Model	BEM-based					AC-based				
	Balanced	Unseen multi-aspect	Unseen comparison	Unseen complex	Unseen open-ended	Balanced	Unseen multi-aspect	Unseen comparison	Unseen complex	Unseen open-ended
Oracle-Router	0.5753	0.3401	0.5217	0.5799	0.5601	0.7948	0.7802	0.7831	0.7978	0.7852
Best Standard RAG	0.3599	0.1941	0.3074	0.3595	0.3465	0.5776	0.5809	0.5723	0.5812	0.5784
`OverallSim`-Router	0.3474	0.1874	0.3004	0.3458	0.3261	0.5722	0.5835	0.5689	0.5668	0.5658
`AvgSim`-Router	0.3542	0.1855	0.2995	0.3454	0.3438	0.5855	0.5871	0.5724	0.5825	0.5739
`MaxSim`-Router	0.3623	0.1913	0.3161	0.3591	0.3521	0.5872	0.5777	0.5753	0.5891	0.5775
`VarSim`-Router	0.3307	0.1864	0.3082	0.3255	0.2941	0.5593	0.5811	0.5553	0.5603	0.5450
`Moran`-Router	0.3304	0.1880	0.3083	0.3255	0.2941	0.5603	0.5840	0.5535	0.5596	0.5462
$XGBoost_{point}$	0.3444	0.1915	0.3065	0.3519	0.3318	0.5760	0.5925	0.5698	0.5790	0.5885
$SVMRank_{point}$	0.3548	0.1894	0.3167	0.3624	0.3284	**0.5903**	0.5815	0.5779	0.5899	0.5885
FFN_{point}	0.3528	0.1894	0.3039	0.3397	0.3211	0.5760	0.5901	0.5650	0.5854	0.5529
$DeBERTa_{point}$	0.3604	0.1911	0.3064	0.3590	0.3387	**0.5884**	0.5847	0.5762	0.5925	0.5863
$XGBoost_{pair}$	0.3626	0.1909	0.3094	0.3625	0.3504	**0.5900**	0.5769	0.5759	**0.6017**	**0.5930**
$\text{SVM}^{rank}{}_{pair}$	0.3657	0.1883	0.2943	0.3643	0.3605	0.5329	0.5842	0.5695	0.5377	0.5791
FFN_{pair}	0.3575	0.1892	0.3015	0.3654	0.3457	0.5831	0.5872	0.5814	0.5798	0.5869
$DeBERTa_{pair}$	0.3634	0.1956	0.3083	0.3654	0.3445	**0.5888**	0.5805	0.5777	0.5912	0.5851
$ListNet_{list}$	0.3530	0.1934	0.3032	0.3398	0.3254	0.5848	0.5836	0.5718	0.5847	0.5818
$LambdaMART_{list}$	0.3450	0.1959	0.2934	0.3631	0.3229	0.5802	0.5812	0.5483	0.5690	0.5662
$DeBERTa_{list}$	0.3235	0.1825	0.3005	0.3470	0.2736	0.5829	0.5902	0.5556	0.5434	0.5751

Table 1: Average downstream RAG utility measured by either BEM or AC when the test queries are routed to the top retriever based on the routing model. Bold: statistically significant improvement over the best standard RAG system according to the paired Wilcoxon signed-rank tests with bonferroni correction.

whether trained routing models (LTRR-based) outperform the highest train-free baselines. As in RQ1, numerical results indicate that LTRR models generally outperform the strongest train-free routers (usually `MaxSim`). However, statistical significance tests revealed that these improvements were not significant after correction. This suggests that the observed gains may be subject to variability and underscores the need for larger-scale studies or refined methods to more conclusively demonstrate the advantages of trained routing over train-free approaches.

RQ3: Are the performance improvements from routing-based RAG models robust across the different unseen query-type splits? We investigated the robustness of performance improvements across various unseen query types (multi-aspect, comparison, complex, open-ended). While train-free routing models showed relatively modest improvements across these splits, LTRR-based trained routing algorithms displayed more stable and consistent performance gains. In particular, the pairwise XGBoost routers trained on the AC utility metric showed the most consistent outperformance over the standard RAG and train-free baselines across different unseen datasets, achieving statistically significant results in the complex and open-ended query splits.

Discussion and Implications. Our findings underscore that not all routing improvements are created equal. While LTRR models often outperform standard and train-free routing methods numerically, only a subset achieve statistically significant gains, particularly those trained with the AC utility metric. This confirms that metric choice is not merely a technical detail, but a determinant of learning signal quality. Moreover, the effectiveness of pairwise training (especially with tree-based models like XGBoost) suggests that explicitly modeling retriever tradeoffs per query offers a more robust inductive bias than listwise or pointwise formulations. The observed performance robustness of LTRR across unseen question types further indicates that routing functions can generalize beyond their training distribution. This points to the potential of query routing as a critical component in adaptive retrieval architectures, especially for long-tailed or evolving query scenarios. Notably, our system is built entirely on lightweight, cost-effective retrievers and a computationally efficient routing model, but still achieves meaningful gains, showing that even modest setups can benefit from query routing. Finally, since LTRR produces a full ranking over retrievers, it naturally supports future extensions to multi-retriever selection, where retrieved results can be fused to enhance coverage and diversity [10]. We leave this extension for future work.

7 Conclusion

We introduce LTRR, a query routing framework that learns to rank retrievers based on their downstream utility to large language models. Our empirical results show that RAG systems equipped with trained routers—particularly those using the AC utility metric and pairwise learning-to-rank algorithms such as XGBoost—can outperform standard single-retriever RAG systems and generalize to unseen query types. These findings highlight the importance of utility-aware retriever selection and demonstrate that learning-based query routing offers a promising path toward more robust and adaptable RAG systems. In the SIGIR 2025 LiveRAG challenge, our team *Starlight* submitted a RAG system equipped with the pairwise XGBoost-based router. The system achieved a Correctness score of 0.818 and a Faithfulness score of 0.433.

Acknowledgments

This work was supported by NSF grant 2402874. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsors.

References

[1] Anthropic. 2024. Introducing the Model Context Protocol. https://www.anthropic.com/news/model-context-protocol Accessed: 2025-05-23.

[2] Jaime Arguello et al. 2017. Aggregated search. *Foundations and Trends® in Information Retrieval* 10, 5 (2017), 365–502.

[3] Jannis Bulian, Christian Buck, Wojciech Gajewski, Benjamin Börschinger, and Tal Schuster. 2022. Tomayto, Tomahto. Beyond Token-level Answer Equivalence for Question Answering Evaluation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang (Eds.). Association for Computational Linguistics, Abu Dhabi, United Arab Emirates, 291–305. doi:10.18653/v1/2022.emnlp-main.20

[4] Jamie Callan. 2002. Distributed information retrieval. In *Advances in information retrieval: recent research from the center for intelligent information retrieval*. Springer, 127–150.

[5] James P Callan, Zhihong Lu, and W Bruce Croft. 1995. Searching distributed collections with inference networks. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*. 21–28.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*. 129–136.

[7] Hsinchun Chen, Haiyan Fan, Michael Chau, and Daniel Zeng. 2001. MetaSpider: Meta-searching and categorization on the Web. *Journal of the American Society for Information Science and Technology* 52, 13 (2001), 1134–1147.

[8] Mingyang Chen, Tianpeng Li, Haoze Sun, Yijie Zhou, Chenzheng Zhu, Haofen Wang, Jeff Z. Pan, Wen Zhang, Huajun Chen, Fan Yang, Zenan Zhou, and Weipeng Chen. 2025. ReSearch: Learning to Reason with Search for LLMs via Reinforcement Learning. arXiv:2503.19470 [cs.AI] https://arxiv.org/abs/2503.19470

[9] Tianqi Chen and Carlos Guestrin. 2016. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 785–794.

[10] Gordon V Cormack, Charles LA Clarke, and Stefan Buettcher. 2009. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. 758–759.

[11] Zhuyun Dai, Yubin Kim, and Jamie Callan. 2017. Learning to rank resources. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*. 837–840.

[12] Fernando Diaz. 2005. Regularizing ad hoc retrieval scores. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. 672–679.

[13] Fernando Diaz. 2007. Performance prediction using spatial autocorrelation. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. 583–590.

[14] Fernando Diaz. 2007. Regularizing query-based retrieval scores. *Information Retrieval* 10 (2007), 531–562.

[15] Fernando Diaz, Mounia Lalmas, and Milad Shokouhi. 2010. From federated to aggregated search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. 910–910.

[16] Fernando Diaz, Bhaskar Mitra, Michael D. Ekstrand, Asia J. Biega, and Ben Carterette. 2020. Evaluating Stochastic Rankings with Expected Exposure. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management (CIKM '20)*. Association for Computing Machinery, 275–284.

[17] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated Evaluation of Retrieval Augmented Generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, Nikolaos Aletras and Orphee De Clercq (Eds.). Association for Computational Linguistics, St. Julians, Malta, 150–158. https://aclanthology.org/2024.eacl-demo.16

[18] Simone Filice, Guy Horowitz, David Carmel, Zohar Karnin, Liane Lewin-Eytan, and Yoelle Maarek. 2025. Generating Diverse Q&A Benchmarks for RAG Evaluation with DataMorgana. arXiv:2501.12789 [cs.CL] https://arxiv.org/abs/2501.12789

[19] Eric J Glover, Steve Lawrence, William P Birmingham, and C Lee Giles. 1999. Architecture of a metasearch engine that supports user information needs. In *Proceedings of the eighth international conference on Information and knowledge management*. 210–216.

[20] Rachid Guerraoui, Anne-Marie Kermarrec, Diana Petrescu, Rafael Pires, Mathis Randl, and Martijn de Vos. 2025. Efficient Federated Search for Retrieval-Augmented Generation. In *Proceedings of the 5th Workshop on Machine Learning and Systems* (World Trade Center, Rotterdam, Netherlands) *(EuroMLSys '25)*. Association for Computing Machinery, New York, NY, USA, 74–81. doi:10.1145/3721146.3721942

[21] Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543* (2021).

[22] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park. 2024. Adaptive-RAG: Learning to Adapt Retrieval-Augmented Large Language Models through Question Complexity. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, Kevin Duh, Helena Gomez, and Steven Bethard (Eds.). Association for Computational Linguistics, Mexico City, Mexico, 7036–7050. doi:10.18653/v1/2024.naacl-long.389

[23] Bowen Jin, Hansi Zeng, Zhenrui Yue, Dong Wang, Hamed Zamani, and Jiawei Han. 2025. Search-r1: Training llms to reason and leverage search engines with reinforcement learning. *arXiv preprint arXiv:2503.09516* (2025).

[24] Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*. 217–226.

[25] Ekaterina Khramtsova, Shengyao Zhuang, Mahsa Baktashmotlagh, Xi Wang, and Guido Zuccon. 2023. Selecting which Dense Retriever to use for Zero-Shot Search. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region* (Beijing, China) *(SIGIR-AP '23)*. Association for Computing Machinery, New York, NY, USA, 223–233. doi:10.1145/3624918.3625330

[26] Ekaterina Khramtsova, Shengyao Zhuang, Mahsa Baktashmotlagh, and Guido Zuccon. 2024. Leveraging LLMs for Unsupervised Dense Retriever Ranking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Washington DC, USA) *(SIGIR '24)*. Association for Computing Machinery, New York, NY, USA, 1307–1317. doi:10.1145/3626772.3657798

[27] To Eun Kim and Fernando Diaz. 2025. Towards Fair RAG: On the Impact of Fair Ranking in Retrieval-Augmented Generation. arXiv:2409.11598 [cs.IR] https://arxiv.org/abs/2409.11598

[28] To Eun Kim, Alireza Salemi, Andrew Drozdov, Fernando Diaz, and Hamed Zamani. 2024. Retrieval-Enhanced Machine Learning: Synthesis and Opportunities. *arXiv preprint arXiv:2407.12982* (2024).

[29] Hyunji Lee, Luca Soldaini, Arman Cohan, Minjoon Seo, and Kyle Lo. 2024. Router-retriever: Exploring the benefits of routing over multiple expert embedding models. *arXiv preprint arXiv:2409.02685* (2024).

[30] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems* 33 (2020), 9459–9474.

[31] Zhiling Luo, Xiaorong Shi, Xuanrui Lin, and Jinyang Gao. 2025. Evaluation Report on MCP Servers. *arXiv preprint arXiv:2504.11094* (2025).

[32] Feiteng Mu, Yong Jiang, Liwen Zhang, Liuchu Liuchu, Wenjie Li, Pengjun Xie, and Fei Huang. 2024. Query Routing for Homogeneous Tools: An Instantiation in the RAG Scenario. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen (Eds.). Association for Computational Linguistics, Miami, Florida, USA, 10225–10230. doi:10.18653/v1/2024.findings-emnlp.598

[33] Harrie Oosterhuis. 2021. Computationally efficient optimization of plackett-luce ranking models for relevance and fairness. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1023–1032.

[34] Ermelinda Oro, Francesco Maria Granata, Antonio Lanza, Amir Bachir, Luca De Grandis, and Massimo Ruffolo. 2024. Evaluating Retrieval-Augmented Generation for Question Answering with Large Language Models. (2024).

[35] Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, et al. 2025. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems* 37 (2025), 30811–30849.

[36] Joseph John Rocchio Jr. 1971. Relevance feedback in information retrieval. *The SMART retrieval system: experiments in automatic document processing* (1971).

[37] Alireza Salemi and Hamed Zamani. 2024. Evaluating retrieval quality in retrieval-augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2395–2400.

[38] Xiaqiang Tang, Jian Li, Nan Du, and Sihong Xie. 2024. Adapting to Non-Stationary Environments: Multi-Armed Bandit Enhanced Retrieval-Augmented Generation on Knowledge Graphs. *arXiv preprint arXiv:2412.07618* (2024).

[39] Falcon-LLM Team. 2024. The Falcon 3 Family of Open Models. https://huggingface.co/blog/falcon3

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

[41] Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. 2023. ColBERT-PRF: Semantic pseudo-relevance feedback for dense passage and document retrieval. *ACM Transactions on the Web* 17, 1 (2023), 1–39.

[42] Qiang Wu, Christopher JC Burges, Krysta M Svore, and Jianfeng Gao. 2010. Adapting boosting for information retrieval measures. *Information Retrieval* 13 (2010), 254–270.

A Reranking Methods

Query-Time Score Regularization. Based on the cluster hypothesis in IR, *query-time score regularization* [12, 14] adjusts retrieval scores so that semantically similar documents receive similar relevance scores. Score regularization can be seen as a generalization of pseudo-relevance feedback (PRF), a well-established method for improving retrieval performance [36, 41]. However, unlike PRF, which modifies query representations, score regularization directly refines retrieval scores. Although shown to improve retrieval performance, it remains underexplored in dense retrieval and especially in a RAG setup.

Reranking procedure is as follows:

(1) **Initial Retrieval:** Pinecone is queried, returning k documents and their n-dim embeddings, $\{d_i\}_{i=1}^{k}$.
(2) **Similarity Matrix:** From a matrix $D \in \mathbb{R}^{k \times n}$, where each row is d_i, we compute a similarity matrix $W = DD^T \in \mathbb{R}^{k \times k}$.
(3) **Row-Stochastic Matrix:** For each row of W, we keep top-m similarities, normalize to sum to 1, yielding a row-stochastic matrix P.
(4) **Regularized Scores:** Given an original score vector $s \in \mathbb{R}^{k \times 1}$, we define $\tilde{s} = P^t s$, where t controls the strength. We then rerank documents by $\tilde{s}$, enhancing clusters of semantically similar passages.

Stochastic Reranking. *Stochastic retrieval* uses Plackett-Luce sampling [33] on the initial score distribution to promote item-fairness [16]. As shown by Kim and Diaz [27], this can boost item-fairness in RAG systems and potentially enhance downstream QA performance. In our reranking, we randomly select one ranking from the 50 sampled rankings, where sampling intensity parameter is set to 2.

B Falcon Prompts

- {INFO_PROMPT} is first constructed by iterating through the retrieved documents by filling in the following template: `Document {DOC_i}: {TEXT}`.
- Fallback prompt is selected when the generator produces an ill-formatted output (e.g., no `<answer>` tag exists).
- Non-RAG prompt is selected when the router selects $\mathcal{R}_0$ and no external information is retrieved.

B.1 RAG Reasoning Prompt

You must answer a user question, based on the information (web documents) provided. Before answering the question, you must conduct your reasoning inside <think> and </think>. During this reasoning step, think about the intent of the user's question and focus on evaluating the relevance and helpfulness of each document to the question. This is important because the information comes from web retrieval and may include irrelevant content. By explicitly reasoning through the relevance and utility of each document, you should seek to ensure that your final answer is accurate and grounded in the pertinent information. After that, think step by step to get to the right answer. Generation format: you need to surround your reasoning with <think> and </think>, and need to surround your answer with <answer> and </answer>. For example: <think> your reasoning </think>; <answer> your answer </answer>.
User Question: {QUESTION}
Information: {INFO_PROMPT}
Show your reasoning between <think> and </think>, and provide your final answer between <answer> and </answer>.

B.2 RAG Fallback Prompt

Using the information provided below, please answer the user's question. Consider the intent of the user's question and focus on the most relevant information that directly addresses what they're asking. Make sure your response is accurate and based on the provided information.
Question: {QUESTION}
Information: {INFO_PROMPT}
Provide a clear and direct answer to "{QUESTION}" based on the information above.

B.3 Non-RAG Reasoning Prompt

You must answer the given user question enclosed within <question> and </question>. Before answering the question, you must conduct your reasoning inside <think> and </think>. During this reasoning step, think about the intent of the user's question then think step by step to get to the right answer. Generation format: you need to surround your reasoning with <think> and </think>, and need to surround your answer with <answer> and </answer>. For example: <think> your reasoning </think>; <answer> your answer </answer>.
User Question: {QUESTION}
Show your reasoning between <think> and </think>, and provide your final answer between <answer> and </answer>.

B.4 Non-RAG Fallback Prompt

Answer the following question with ONLY the answer. No explanations, reasoning, or additional context.
Question: {QUESTION}

C LTRR Loss Functions

Pointwise Approach. Each retriever's utility gain δ_i is independently predicted by minimizing a regression loss:

$$L_{point}(f) = \sum_{q \in Q} \sum_{i} (\delta_i - f(\Phi(q, \mathcal{R}_i)))^2. \tag{4}$$

Pairwise Approach. The model learns from pairwise comparisons between retrievers, minimizing ranking inversions through a pairwise loss, where ranking preference ($\mathcal{R}_i \succ_q \mathcal{R}_j$) is determined based on the relative utility gain each retriever provides to the RAG system compared to the no-retrieval baseline:

$$L_{pair}(f) = \sum_{q \in Q} \sum_{\mathcal{R}_i \succ_q \mathcal{R}_j} \mathrm{I}\left[f(\Phi(q, \mathcal{R}_i)) < f(\Phi(q, \mathcal{R}_j))\right]. \tag{5}$$

Listwise Approach. The model directly optimizes the predicted utility gains across all retrievers for each query. With P_i and Q_i being the probability distribution over rankings based on utility-gain and model prediction, we define the listwise loss as:

$$L_{list}(f) = \sum_{q \in Q} \sum_{i} -P_i(\delta_i) log(Q_i(f(\Phi(q, \mathcal{R}_i)))). \tag{6}$$

D Dataset Generation via DataMorgana

Table 2 describes question type configurations, and Table 3 describes user type configuration for dataset generation using DataMorgana [18].

Categorization	Category	Prob	Description
Answer Type	factoid	0.4	a question seeking a specific, concise piece of information or a fact about a particular subject.
	multi-aspect	0.2	A question about two different aspects of the same entity/concept. For example: 'What are the advantages of AI-powered diagnostics, and what are the associated risks of bias in medical decision-making?', 'How do cryptocurrencies enable financial inclusion, and what are the security risks associated with them?'. The information required to answer the question needs to come from two documents, specifically, the first document must provide information about the first aspect, while the second must provide information about the second aspect.
	comparison	0.2	a comparison question that requires comparing two related concepts or entities. The comparison must be natural and reasonable, i.e., comparing two entities by a common attribute which is meaningful and relevant to both entities. For example: 'Who is older, Glenn Hughes or Ross Lynch?', 'Are Pizhou and Jiujiang in the same province?', 'Pyotr Ilyich Tchaikovsky and Giuseppe Verdi have this profession in common'. The information required to answer the question needs to come from two documents, specifically, the first document must provide information about the first entity/concept, while the second must provide information about the second entity/concept.
	open-ended	0.2	a question seeking a detailed or exploratory response, encouraging discussion or elaboration.
Premise	without-premise	0.7	a question that does not contain any premise or any information about the user.
	with-premise	0.3	a question starting with a very short premise, where the user reveals one's needs or some information about himself.
Phrasing	concise-and-natural	0.25	a concise, direct, and natural question consisting of a few words.
	verbose-and-natural	0.25	a relatively long question consisting of more than 9 words.
	short-search-query	0.25	a question phrased as a typed web query for search engines (only keywords, without punctuation and without a natural-sounding structure). It consists of less than 7 words.
	long-search-query	0.25	a question phrased as a typed web query for search engines (only keywords without punctuation and without a natural-sounding structure). It consists of more than 6 words.
Linguistic Variation	similar-to-document	0.5	a question that is written using the same or similar terminology and phrases appearing in the documents.
	distant-from-document	0.5	a question that is written using the terms completely different from the ones appearing in the documents.

Table 2: Question categorizations and descriptions.

Categorization	Category	Prob	Description
User Expertise	expert	0.4	an expert on the subject discussed in the documents, therefore he asks complex questions.
	novice	0.6	a person with basic knowledge on the topic discussed in the documents, therefore, he asks non-complex questions.

Table 3: User categorization and descriptions.

E Data Statistics

Table 4 shows statistics of training and test data from the DataMorgana-generated dataset.

	Train	Test
Balanced	7,995	1999
multi-aspect	6442	410
comparison	6457	385
complex	4751	787
open-ended	6345	411

Table 4: The number of queries in each dataset for training and testing LTRR algorithms.

F Standard RAG Performance

Table 5 shows the average downstream utilities of standard RAG models. Models with the highest utility value were selected as the 'Best Standard RAG' models in Table 1.

Model	BEM-based					AC-based				
	Balanced	Unseen multi-aspect	Unseen comparison	Unseen complex	Unseen open-ended	Balanced	Unseen multi-aspect	Unseen comparison	Unseen complex	Unseen open-ended
BM25	**0.3599**	0.1826	0.3034	**0.3595**	0.3430	0.5769	0.5656	0.5706	**0.5812**	0.5711
BM25+Stochastic	0.3302	0.1796	0.3046	0.3252	0.3042	0.5510	0.5688	0.5469	0.5543	0.5424
BM25+Regularize	0.3539	0.1893	0.3061	0.3559	0.3350	**0.5776**	0.5696	**0.5723**	0.5790	0.5749
E5	0.3538	**0.1941**	**0.3074**	0.3496	**0.3465**	0.5769	0.5791	0.5711	0.5782	**0.5784**
E5+Stochastic	0.3258	0.1855	0.3049	0.3252	0.3002	0.5503	0.5805	0.5483	0.5417	0.5546
E5+Regularize	0.3418	0.1829	0.2991	0.3490	0.3083	0.5694	**0.5809**	0.5692	0.5756	0.5620

Table 5: Average downstream utility measured by either BEM or AC of standard RAG systems with a single retriever model. Boldface indicates the highest average downstream utility value.