Efficient Federated Search for Retrieval-Augmented Generation

Rachid Guerraoui
EPFL
Lausanne, Switzerland

Anne-Marie Kermarrec
EPFL
Lausanne, Switzerland

Diana Petrescu*
EPFL
Lausanne, Switzerland

Rafael Pires
EPFL
Lausanne, Switzerland

Mathis Randl
EPFL
Lausanne, Switzerland

Martijn de Vos
EPFL
Lausanne, Switzerland

Abstract

Large language models (LLMs) have demonstrated remarkable capabilities across various domains but remain susceptible to hallucinations and inconsistencies, limiting their reliability. Retrieval-augmented generation (RAG) mitigates these issues by grounding model responses in external knowledge sources. Existing RAG workflows often leverage a single vector database, which is impractical in the common setting where information is distributed across multiple repositories. We introduce RAGRoute, a novel mechanism for federated RAG search. RAGRoute dynamically selects relevant data sources at query time using a lightweight neural network classifier. By not querying every data source, this approach significantly reduces query overhead, improves retrieval efficiency, and minimizes the retrieval of irrelevant information. We evaluate RAGRoute using the MIRAGE and MMLU benchmarks and demonstrate its effectiveness in retrieving relevant documents while reducing the number of queries. RAGRoute reduces the total number of queries up to 77.5% and communication volume up to 76.2%.

CCS Concepts: • **Information systems** → **Retrieval models and ranking**; **Combination, fusion and federated search**; • **Computing methodologies** → **Natural language processing**.

Keywords: Retrieval-Augmented Generation, Large Language Models, Federated Search, Resource Selection, Routing

*Corresponding author: diana.petrescu@epfl.ch

EuroMLSys '25, March 30-April 3 2025, Rotterdam, Netherlands

ACM ISBN 979-8-4007-1538-9/2025/03
https://doi.org/10.1145/3721146.3721942

ACM Reference Format:
Rachid Guerraoui, Anne-Marie Kermarrec, Diana Petrescu, Rafael Pires, Mathis Randl, and Martijn de Vos. 2025. Efficient Federated Search for Retrieval-Augmented Generation. In *The 5th Workshop on Machine Learning and Systems (EuroMLSys '25), March 30-April 3 2025, Rotterdam, Netherlands.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3721146.3721942

1 Introduction

Large language models (LLMs) have driven significant advancements across various industries and domains [4], including natural language processing [19], healthcare [12] and decision support systems [10]. Despite their widespread adoption, it remains difficult to ensure the reliability of their outputs [28]. One major concern is their tendency to *hallucinate*, generating false or misleading responses with high confidence, which diminishes their usability in critical applications [15]. Additionally, their responses can vary *inconsistently* across queries, particularly in specialized or intricate domains, often requiring the verification of responses by domain experts [21].

A well-known method for enhancing the reliability of LLM responses is to use retrieval-augmented generation (RAG) [22]. RAG integrates LLM text generation with external information retrieval, enabling models to ground their responses in credible sources. This approach first retrieves relevant documents from an external database and then incorporates them into the LLM prompt before response generation. By leveraging external knowledge sources, RAG allows LLMs to provide more accurate and contextually relevant answers without requiring model parameter updates through compute-expensive retraining or fine-tuning [26].

RAG workflows typically query embeddings from a single monolithic vector database [20]. However, in many industries, it is natural for information to be scattered across multiple repositories [5]. Medical professionals, for example, need to retrieve patient records, clinical guidelines, and recent research findings from multiple information systems [16]. Similarly, legal professionals must consult multiple independent sources to build a case or provide legal advice. This calls for *federated RAG search*, which is essentially a mechanism that can query multiple data sources and aggregate relevant

information [29]. Such a mechanism has multiple advantages over using a single database. Firstly, it sidesteps the need to move data to a central database, which might be complicated due to regulatory constraints [18]. Secondly, it allows for seamless extension of existing databases, without requiring data migration or duplication across sites, since data is represented by high-dimensional vectors, or *embeddings*. Thirdly, it ensures that organizations can reuse their existing infrastructure while enabling users to query multiple sources efficiently, reducing storage overhead and maintaining data consistency.

The effectiveness of federated RAG search depends on a resource selection mechanism that decides which data stores are most likely to contain relevant documents [31]. Without such a mechanism, a RAG system would query all available data sources indiscriminately, leading to several problems: *(i)* obtaining information from irrelevant sources might increase the chances for LLMs to hallucinate and reduce the quality of the generated responses [31]; and *(ii)* the overhead, in terms of communication volume and computational cost of retrieving embeddings from every possible source can be significant. This overhead becomes particularly problematic in large-scale deployments, where response times and cost-effectiveness are critical.

This work introduces RAGRoute, a new routing mechanism that enables federated RAG search by dynamically selecting relevant sources at query time. RAGRoute is powered by a lightweight neural network classifier. RAGRoute first trains a neural network classifier on the characteristics of available data sources, which is then used to route subsequent queries efficiently. This approach significantly reduces the number of accessed nodes, thereby lowering resource consumption while maintaining high retrieval quality. Thus, RAGRoute proactively learns a routing policy tailored to the structure of the data sources and the nature of queries.

We evaluate the effectiveness and efficiency of RAGRoute using two standard benchmarks: MIRAGE and MMLU. RAGRoute achieves high retrieval recall and shows excellent performance in determining whether data sources are relevant for a given query. Notably, RAGRoute reduces the total number of queries up to 77.5% and communication volume up to 76.2%.

Our contributions are as follows:

1. We introduce RAGRoute, a novel and efficient approach for federated RAG search (Section 3). At the core of RAGRoute lies a lightweight neural network classifier that determines the relevance of each data source and routes subsequent queries accordingly. This sidesteps the need for a given user query to contact all, possibly irrelevant data sources.
2. We implement RAGRoute and conduct an experimental evaluation (Section 4). Our evaluation with two standard benchmarks demonstrates that RAGRoute is both effective and efficient, underscoring the potential of our approach for federated RAG search.

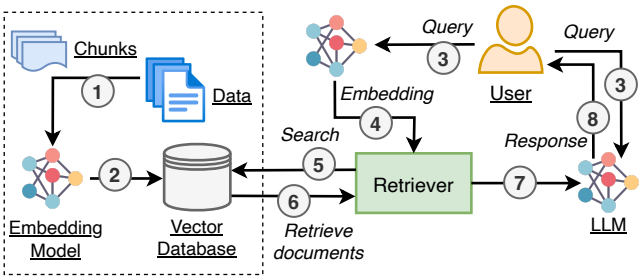

Figure 1. The RAG workflow.

2 Background and problem description

We first explain the RAG workflow and how RAG uses vector databases. We then introduce the concept of federated search and its associated challenges.

2.1 Retrieval-augmented generation (RAG)

RAG enhances the reliability of LLM responses by integrating retrieved information as part of the input (or *prompt*) [22]. Unlike traditional models that rely solely on pre-trained knowledge stored in their weights, RAG retrieves relevant external documents or data during inference, improving accuracy and reducing hallucinations. This approach is particularly helpful for tasks requiring up-to-date information or factual knowledge.

We illustrate the RAG workflow in Figure 1 which consists of eight steps. At first, documents, videos, or other data are split into chunks. Each chunk is then converted into a high-dimensional vector using an embedding model (step 1). These embeddings are then stored in a vector database (2). When a user submits a query (3), it is transformed into an embedding and passed to the retriever (4), which searches for the most relevant stored embeddings (5) and retrieves the corresponding data chunks (6). The retrieved context and the original query are then fed into the LLM, which generates a response grounded in the retrieved information (7). This enriched query ultimately returns a more accurate and context-aware answer to the user than when not using RAG (8).

The vector database is a specialized database designed for similarity search, where queries are matched based on their vector representations rather than exact keyword matches. In the context of RAG, each document in the database is first converted into a high-dimensional embedding using a pre-trained machine learning (ML) model. By leveraging embeddings, vector databases can capture the semantic similarity between queries and documents, allowing for more flexible and context-aware retrieval compared to traditional keyword-based search methods [13]. When a query is issued,

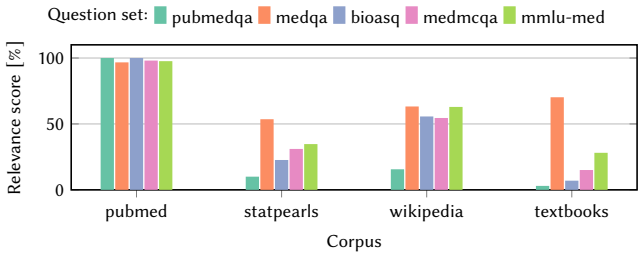

Figure 2. The relevance of different corpora in RAG when answering questions, using question sets from MIRAGE.

it is similarly transformed into an embedding, and the database retrieves the most relevant documents by performing a nearest-neighbor search in the embedding space. This search is typically accelerated using indexing techniques such as approximate nearest neighbor (ANN) [23], enabling efficient retrieval even with large data sources.

2.2 Towards federated RAG search

Federated search is a retrieval paradigm where a query is executed across multiple independent data sources, aggregating relevant results without requiring data to be present in a single database [29]. In the context of RAG, federated search can enhance response accuracy by dynamically selecting appropriate knowledge bases for a given query.

A key challenge in federated RAG search is determining which data sources are relevant to a given query and retrieving information only from those sources. Not all data sources contribute equally to the queries. We empirically show this by analyzing chunk relevance using corpora and questions from the MIRAGE benchmark. Figure 2 shows the relevance of different corpora, highlighting the variability in corpus relevance depending on the query. For example, the bar corresponding to the MedQA question set and the StatPearls corpus shows a relevance score of 53.6%, meaning that for 53.6% of queries in MedQA, at least one chunk in the retrieved relevant chunks originates from StatPearls. While some corpora, such as PubMed, consistently provide valuable information for all question sets, relying on a single corpus is often insufficient. Indeed, results from [34] demonstrate that combining multiple corpora improves retrieval performance. Corpora, such as StatPearls or Wikipedia, are useful in specific cases. The differences in corpus relevance underscore the importance of adequate resource selection for a given query.

One must strike a balance in the number of data sources being queried. Over-selecting data sources can dilute relevance by introducing potentially irrelevant data chunks in the LLM prompt, making it harder for the LLM to extract useful knowledge. Under-selecting data sources risks missing critical information, particularly in domains where information is distributed sparsely across multiple repositories. At the same time, we aim for the overall retrieval latency to be as low as possible, ensuring timely LLM responses. Achieving a good trade-off between retrieval efficiency and response quality remains an open problem. Therefore, this work answers the following question: *how can we design an efficient routing mechanism for federated RAG search that minimize retrieval overhead while selecting relevant data sources?*

3 Design of RAGRoute

We now introduce the design of RAGRoute. In the following, we assume that there are n data sources that can contain information relevant to a user query. First, we explain the high-level workflow of RAGRoute and then provide details on how RAGRoute selects which data sources to query.

3.1 RAGRoute workflow

We visualize the RAGRoute workflow in Figure 3, allowing the querying of n distinct data sources. These data sources can, for example, be distributed across different organizations. We show the components specific to RAGRoute in the dashed box. In line with existing RAG systems, the query by a user (step 1) is converted into an embedding using an embedding model (2). However, this query embedding is then forwarded to a *router*, whose purpose is to decide which of the n data sources are relevant. We detail the design of our router in Section 3.2.

After determining the relevant data sources, we send a query to these data sources with the vector embedding. Using top-k querying with n data sources results in $m \times k$ retrieved embeddings, where m is the number of contacted data sources ($m \leq n$) (4). We refine the results by selecting the k embeddings closest to the query embedding (5). Using these embeddings, we retrieve the associated data chunks (6). Finally, the original user query and relevant data chunks are fed to the LLM (7), and a response is generated (8).

3.2 Lightweight query routing with shallow neural networks

To enable efficient retrieval across multiple data sources, RAGRoute uses a lightweight query router, implemented as a shallow neural network (NN) with several fully connected layers. This suffices to determine the relevance of each data source before retrieval. Using a shallow NN is inspired by practices in mixture of experts (MoE) models and ensembles. MoE models leverage a small router function to decide which subset of experts to activate [36]. Similarly, shallow neural networks are used for decision-making in one-shot federated ensembles [2]. This work applies similar ideas to selecting relevant data sources in RAG.

3.2.1 Training phase. The RAGRoute router learns to make routing decisions by looking at query-data source relevance, essentially replaying step 3-5 in Figure 3 for model

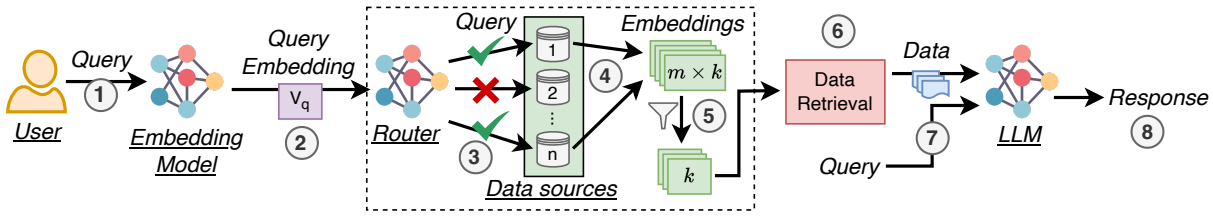

Figure 3. The workflow of RAGRoute. The components specific to RAGRoute are indicated in the box with the dashed border. In contrast to existing RAG workflows that rely on a single data store, RAGRoute enables efficient federated search by using a lightweight router to determine relevant data sources during an inference request.

training. Specifically, we send a set of training query embeddings to all n data sources to obtain relevant embeddings and reduce the obtained $n \times k$ embeddings to the top-k most relevant ones, *e.g.*, based on their distance to the query embedding. k is a parameter chosen by the user that we use for training and inference. RAGRoute assigns a binary relevance indicator to each query-data source pair based on whether or not the merged top-k results contain embeddings from a given data source. The model takes the following five features as input: *(i)* the query embedding, *(ii)* the centroid of the queried data source, *(iii)* the distance between the query embedding and the centroid, *(iv)* the number of items in the queried data source, and *(v)* the density of the queried data source. The centroid, computed as the average vector representation of all document embeddings in a data source, summarizes its overall semantic content. The density of the data source quantifies how tightly packed the document embeddings are around the centroid, indicating whether the data source is highly specialized (high density) or more diverse (low density). A single routing model is then trained using this data and the labels, allowing it to predict whether a given data source is relevant for future queries.

3.2.2 Inference phase. Once trained, RAGRoute uses this model to efficiently route incoming user queries to relevant data sources. We do an inference for each of the available data sources. The inference request for each data source completes quickly (with sub-millisecond latency, see Section 4.4) and can be done in parallel or in batches since they are independent.

4 Evaluation

We implement RAGRoute and evaluate its effectiveness and efficiency [1]. Specifically, we measure *(i)* the effectiveness of our routing mechanism in selecting relevant data sources (Section 4.3), *(ii)* the reduction in the total number of queries and communication volume (Section 4.4), and *(iii)* the impact on end-to-end RAG accuracy compared to querying all available sources (Section 4.5).

[1] https://github.com/sacs-epfl/ragroute.

4.1 Experimental setup

We next outline the details of the router model and datasets used in our evaluation.

4.1.1 Router model. We use a three-layer fully connected NN for routing. The network consists of two hidden layers: the first hidden layer has 256 neurons, followed by Layer Normalization, ReLU activation, and Dropout, while the second hidden layer has 128 neurons, also followed by Layer Normalization, ReLU activation, and Dropout. The output layer consists of a single neuron that produces a raw logit score, predicting whether the corpus is relevant to the given query. The model is trained using Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) with a positional weight to address class imbalance. We use a cyclic scheduler for the learning rate γ, oscillating γ between 0.001 and 0.005. Model performance is evaluated on the validation set after each epoch, and the best model is selected based on validation accuracy. To train the model, we split training data at the question level, with a train-validation-test split ratio of 30%-10%-60%. Features are standardized using StandardScaler to normalize input distributions.

4.1.2 Datasets. We evaluate RAGRoute with two commonly used benchmarks: MIRAGE [34] and MMLU [14].

MIRAGE is a benchmark designed to systematically evaluate RAG systems for medical question answering [34]. It consists of 7663 questions drawn from five widely used medical QA datasets. We use MedRAG as knowledge source, which includes five corpora with data chunks related to healthcare [34]. For generating embeddings, we use MedCPT [17], a domain-specific model designed for biomedical contexts. For retrieval, we use the IndexFlatL2 index structure, provided by the FAISS library [8], ensuring exact search and eliminating sources of approximation in our experiments. We treat each corpus as a separate data source. To emulate the setting with a single data source (conventional RAG) in some experiments, all corpora are combined into a single dataset, referred to as MedCorp. To run RAGRoute with a RAG pipeline, we adopt and extend the MedRAG toolkit. As LLM, we use the open-source LLaMA 3.1 Instruct model [9], and use the Ollama framework for inference [25].

Experiment	Accuracy (%)	Recall (%)	F1-Score (%)	AUC (%)
MIRAGE (Top 32)	85.63 ± 3.92	85.47 ± 3.61	85.79 ± 2.45	92.6 ± 2.33
MIRAGE (Top 10)	87.3 ± 6.1	88.32 ± 3.96	85.43 ± 4.18	93.67 ± 3.33
MMLU (Top 10)	90.06 ± 5.04	76.23 ± 6.64	78.29 ± 7.59	92.88 ± 3.29

Table 1. Classification metrics (averages and standard deviations) for our router and for different experiments.

MMLU is a benchmark that evaluates LLM systems across tasks ranging from elementary mathematics to legal reasoning [14]. We use ten subject-specific subsets of MMLU for our experiments, with a total of 2313 questions. As a knowledge source, we use a subset of the Wikimedia/Wikipedia dataset, sourced from Wikipedia and embedded using the Cohere Embed V3 model [6]. From this dataset, we extract one million snippets and cluster them into ten groups using the k-means algorithm to simulate different data sources. After clustering, we observe significant variance in the cluster size, ranging from 49 397 to 148 341 snippets per cluster. We use the same LLM as for the MIRAGE benchmark. To run MMLU, we use the RQABench framework [30].

4.2 Hardware

We run our experiments on our university cluster[2]. Each node has a NVIDIA A100 GPU and contains 500 GB of main memory.

4.3 RAGRoute routing effectiveness

We first explore the effectiveness of RAGRoute routing, regarding retrieval recall and classification performance.

Retrieval recall. Figure 4 shows the mean recall for MIRAGE (top 10 and top 32) and MMLU. For MIRAGE, we show recall for different corpora and question sets; for MMLU, we show recall for each data source. Recall is computed by comparing the retrieved snippets for a question against the set of relevant snippets across all data sources, to establish a ground truth. The recall value indicates the proportion of relevant chunks retrieved by each corpus when queried independently. We also show the recall of RAGRoute.

Figure 4 shows that for MIRAGE, PubMed achieves high recall across all benchmarks, while other corpora show variable recall performance. No single corpus can reliably cover all queries. For MMLU, the eighth cluster achieves a mean recall of 49.4%, whereas all other clusters have a recall below 15.3%. RAGRoute consistently achieves high recall across all benchmarks: for MIRAGE, this is between 95.3% and 99.0% for top-10 retrieval and between 96.7% and 98.5% for top-32 retrieval. For MMLU, we observe 90% recall overall. Thus, our router is effective at retrieving relevant data chunks.

Classification performance. To further evaluate the effectiveness of our router, we show its classification performance in predicting corpora relevance for a given query. Table 1 presents various classification metrics, *i.e.*, accuracy, precision, recall, F1-score, and AUC, for our three experiments. In this context, recall refers to the performance of the routing model in identifying relevant corpora, rather than the recall of retrieved snippets within the retrieved context. Similarly, accuracy refers to the classification accuracy of the routing model, not the end-to-end LLM accuracy in generating final responses.

We achieve relatively high accuracy for all experiments, ranging from 85.6% for MIRAGE (Top 32) to 90.1% for MMLU. The accuracy and recall for top-10 retrieval compared to top-32 retrieval are slightly lower, possibly because this is a more difficult classification problem. Finally, we observe that recall and F1-score for MMLU are lower than those for MIRAGE, likely because relevant data sources in MMLU cover a broader range of topics, making them less distinct compared to the more structured and domain-specific nature of MIRAGE's medical dataset. Nevertheless, Table 1 shows that our router is effective at determining relevant data sources.

4.4 RAGRoute efficiency gains

Next, we quantify the reduction by RAGRoute in the number of queries and communication volume.

Number of queries. Figure 5 shows the number of queries for both benchmarks when querying all data sources (naive), when querying relevant data sources (assuming ground truth knowledge), and when using RAGRoute (predicted). We show these results for both benchmarks and for top-10 and top-32 retrieval for MIRAGE. Figure 5 demonstrates that the number of queries sent by RAGRoute is significantly lower compared to querying all data sources. In the top-32 retrieval setting and for MIRAGE, the reduction in queries sent to data sources ranges from 28.93% for MedQA to 69.5% for PubMedQA. Routing achieves even greater efficiency gains in top-10 retrieval settings, with reductions for MIRAGE ranging from 39.9% for MedQA to 71.3% for PubMedQA. For MMLU, the reduction of RAGRoute compared to naive routing is 77.5%, reducing the number of queries from 13 890 to 3126. We further observe that the number of queries by RAGRoute approximates the optimal routing scenario, highlighting the effectiveness of our solution.

Communication volume. RAGRoute also decreases communication volume. For top-32 retrieval and MIRAGE, this reduction ranges from 22.3% for MedQA (200.0 MB → 155.4 MB) to 53.70% for PubMedQA (622.18 MB → 341.04 MB). In this setting, aggregating across all question banks, routing reduces total data transfer by 41.13%, decreasing communication volume from 1156.34 MB under the naive setting to 680.72 MB. The gains in efficiency for the MMLU benchmark are more pronounced due to the increased number of data sources, leading to a 76.2% reduction in communication volume from 73.3 MB to merely 17.42 MB.

Inference time. The routing inference time is minimal in terms of latency. Inference with a batch size of 32 completes

[2]See https://www.epfl.ch/research/facilities/rcp/.

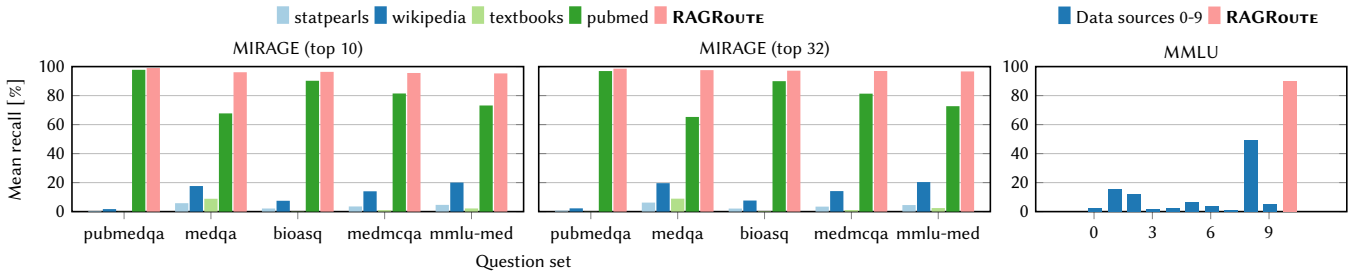

Figure 4. The mean recall for both benchmarks and for different data sources. We also show the mean recall for RAGRoute.

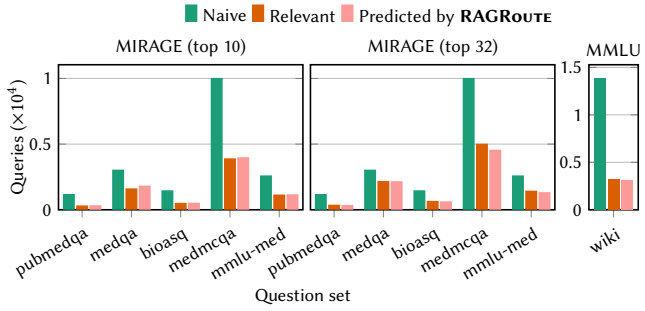

Figure 5. The number of queries for both benchmarks and for different routing strategies.

Corpus	Top 32 Accuracy (%)	Top 10 Accuracy (%)
No RAG	67.04 ± 7.66	67.04 ± 7.66
RAG (all corpora)	72.22 ± 9.86	72.21 ± 10.33
RAGRoute (our work)	**72.24 ± 9.36**	**72.00 ± 10.57**

Table 2. Mean and standard deviation of end-to-end RAG accuracy for configurations with the MIRAGE benchmark.

within 0.3 milliseconds with an NVIDIA A100 GPU and 0.8 milliseconds with a AMD EPYC 7543 32-Core CPU. As such, the overhead of routing has a negligible impact on the end-to-end latency of queries. Because our router is lightweight, it also suitable for usage on low-resource devices.

4.5 End-to-end RAG accuracy

Finally, we compute the average end-to-end accuracy of MIRAGE across top-10 and top-32 retrieval settings for different corpora and for MMLU. The results for MIRAGE are shown in Table 2. Without RAG, we achieve a 67.0% accuracy. When using traditional RAG with a single database containing all corpora, this accuracy increases to 72.22% and 72.21% for top-10 and top-32 retrieval, respectively, surpassing all individual corpora. We observe that using individual corpora, such as StatPearls, can lead to a decrease in accuracy compared to not using RAG (not shown in Table 2). This occurs because the information from a single corpus is not always beneficial for RAG, as also shown in Figure 2. When using RAGRoute, we achieve 72.24% and 72.0% accuracy. For the MMLU benchmark, we get 43.59% accuracy using a single database with all the data and 43.29% accuracy using RAGRoute. Thus, RAGRoute only has a marginal impact on achieved RAG accuracy. These results further reinforce that querying all corpora is not necessary for achieving high accuracy.

5 Related work

RAG with multiple data sources. Expanding the RAG workflow to support multiple data sources is an emerging area of research. FeB4RAG examines federated search within the RAG paradigm and focuses on optimizing resource selection and result merging to enhance retrieval efficiency [31]. The underlying idea consists of introducing a dataset for federated search and incorporating LLM-based relevance judgments to benchmark resource selection strategies. Notably, the paper emphasizes the importance of developing novel federated search strategies to enhance RAG effectiveness. Salve et al. propose a multi-agent RAG system where different agents handle the querying of databases with differing data formats (*e.g.*, relational or NoSQL) [27].

Other approaches focus on privacy when querying data sources across organizations. Raffle is a framework that integrates RAG into the federated learning pipeline and leverages public datasets during training while using private data only at inference time [24]. C-FedRag is a federated RAG approach that enables queries across multiple data sources and leverages hardware-based trusted execution environments (TEEs) to ensure data confidentiality [1]. FRAG leverages homomorphic encryption to enable parties to collaboratively perform ANN searches on encrypted query vectors and data stored in distributed vector databases, ensuring that no party can access others' data or queries [35]. These schemes can benefit from RAGRoute while ensuring privacy-preserving federated search.

ML-assisted resource selection. ML models have been explored to support resource selection in federated search [11]. Wang et al. propose a LLM fine-tuning method that predicts the relevance of previously logged queries and snippets from resources [32]. Arguello et al. leverage different features, *e.g.*,

the topic of queries, and train a classifier for resource selection [3]. Learn-to-rank approaches such as SVMRank [7] and the LambdaMART-based LTRRS [33] refine relevance rankings by leveraging diverse feature sets. However, these models typically are more computationally expensive than the lightweight router used in RAGRoute.

6 Conclusion

We introduced RAGRoute, a novel and efficient routing mechanism for federated RAG. By dynamically selecting relevant data sources at query time using a lightweight neural network classifier, RAGRoute reduces query overhead while maintaining high retrieval quality. Experiments using the MIRAGE and MMLU benchmarks showed that RAGRoute achieves high retrieval recall and superior resource efficiency compared to querying all data sources. Depending on the benchmark, our approach reduces the total number of queries by up to 77.5% and communication volume by up to 76.2%. Our results confirm that querying all data sources is often unnecessary, underscoring the importance of query-aware retrieval strategies in RAG workflows.

Acknowledgments

This work has been funded by the Swiss National Science Foundation, under the project "FRIDAY: Frugal, Privacy-Aware and Practical Decentralized Learning", SNSF proposal No. 10.001.796.

References

[1] Parker Addison, Minh-Tuan H Nguyen, Tomislav Medan, Mohammad T Manzari, Brendan McElrone, Laksh Lalwani, Aboli More, Smita Sharma, Holger R Roth, Isaac Yang, et al. C-fedrag: A confidential federated retrieval-augmented generation system. *arXiv preprint arXiv:2412.13163*, 2024.

[2] Youssef Allouah, Akash Dhasade, Rachid Guerraoui, Nirupam Gupta, Anne-Marie Kermarrec, Rafael Pinot, Rafael Pires, and Rishi Sharma. Revisiting ensembling in one-shot federated learning. *Advances in Neural Information Processing Systems*, 37:68500–68527, 2025.

[3] Jaime Arguello, Jamie Callan, and Fernando Diaz. Classification-based resource selection. In *Proceedings of the 18th ACM conference on Information and knowledge management*, pages 1277–1286, 2009.

[4] G Bharathi Mohan, R Prasanna Kumar, P Vishal Krishh, A Keerthinathan, G Lavanya, Meka Kavya Uma Meghana, Sheba Sulthana, and Srinath Doss. An analysis of large language models: their impact and potential applications. *Knowledge and Information Systems*, pages 1–24, 2024.

[5] Suresh K Bhavnani and Concepción S Wilson. Information scattering. *Encyclopedia of library and information sciences*, pages 2564–2569, 2009.

[6] Cohere. Wikipedia 2023-11 embed multilingual v3, 2023. Accessed: 2025-02-10.

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

[8] Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. 2024.

[9] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[10] Eva Eigner and Thorsten Händler. Determinants of llm-assisted decision-making. *arXiv preprint arXiv:2402.17385*, 2024.

[11] Adamu Garba, Shengli Wu, and Shah Khalid. Federated search techniques: an overview of the trends and state of the art. *Knowledge and Information Systems*, 65(12):5065–5095, 2023.

[12] Joschka Haltaufderheide and Robert Ranisch. The ethics of chatgpt in medicine and healthcare: a systematic review on large language models (llms). *NPJ digital medicine*, 7(1):183, 2024.

[13] Yikun Han, Chunjiang Liu, and Pengfei Wang. A comprehensive survey on vector database: Storage and retrieval technique, challenge. *arXiv preprint arXiv:2310.11703*, 2023.

[14] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

[15] Ziwei Ji, Tiezheng Yu, Yan Xu, Nayeon Lee, Etsuko Ishii, and Pascale Fung. Towards mitigating llm hallucination via self reflection. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 1827–1843, 2023.

[16] Emily Jiang. *Clinical Question-Answering over Distributed EHR Data*. PhD thesis, Massachusetts Institute of Technology, 2024.

[17] Qiao Jin, Won Kim, Qingyu Chen, Donald C. Comeau, Lana Yeganova, W. John Wilbur, and Zhiyong Lu. MedCPT: Contrastive pre-trained transformers with large-scale pubmed search logs for zero-shot biomedical information retrieval. *Bioinformatics*, 2023.

[18] Peter Kairouz, H Brendan McMahan, Brendan Avent, Aurélien Bellet, Mehdi Bennis, Arjun Nitin Bhagoji, Kallista Bonawitz, Zachary Charles, Graham Cormode, Rachel Cummings, et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning*, 14(1–2):1–210, 2021.

[19] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[20] Sanjay Kukreja, Tarun Kumar, Vishal Bharate, Amit Purohit, Abhijit Dasgupta, and Debashis Guha. Performance evaluation of vector embeddings with retrieval-augmented generation. In *2024 9th International Conference on Computer and Communication Systems (ICCCS)*, pages 333–340. IEEE, 2024.

[21] Yoonjoo Lee, Kihoon Son, Tae Soo Kim, Jisu Kim, John Joon Young Chung, Eytan Adar, and Juho Kim. One vs. many: Comprehending accurate information from multiple erroneous and inconsistent ai generations. In *The 2024 ACM Conference on Fairness, Accountability, and Transparency*, pages 2518–2531, 2024.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020.

[23] Wen Li, Ying Zhang, Yifang Sun, Wei Wang, Mingjie Li, Wenjie Zhang, and Xuemin Lin. Approximate nearest neighbor search on high dimensional data—experiments, analyses, and improvement. *IEEE Transactions on Knowledge and Data Engineering*, 32(8):1475–1488, 2019.

[24] Aashiq Muhamed, Pratiksha Thaker, Mona T Diab, and Virginia Smith. Cache me if you can: The case for retrieval augmentation in federated learning. In *Privacy Regulation and Protection in Machine Learning*.

[25] Ollama. Ollama: Get up and running with large language models. GitHub repository, 2025. Accessed: February 8, 2025.

[26] Tolga Şakar and Hakan Emekci. Maximizing rag efficiency: A comparative analysis of rag methods. *Natural Language Processing*, 31(1):1–25, 2025.

[27] Aniruddha Salve, Saba Attar, Mahesh Deshmukh, Sayali Shivpuje, and Arnab Mitra Utsab. A collaborative multi-agent approach to

retrieval-augmented generation across diverse data. *arXiv preprint arXiv:2412.05838*, 2024.

[28] Xiang Shi, Jiawei Liu, Yinpeng Liu, Qikai Cheng, and Wei Lu. Know where to go: Make llm a relevant, responsible, and trustworthy searchers. *Decision Support Systems*, 188:114354, 2025.

[29] Milad Shokouhi, Luo Si, et al. Federated search. *Foundations and Trends® in Information Retrieval*, 5(1):1–102, 2011.

[30] MyScale Team. Retrieval-qa-benchmark: A benchmark for evaluating retrieval-augmented qa systems. GitHub repository, 2024. Accessed: 2025-02-11.

[31] Shuai Wang, Ekaterina Khramtsova, Shengyao Zhuang, and Guido Zuccon. Feb4rag: Evaluating federated search in the context of retrieval augmented generation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 763–773, 2024.

[32] Shuai Wang, Shengyao Zhuang, Bevan Koopman, and Guido Zuccon. Resllm: Large language models are strong resource selectors for federated search. *arXiv preprint arXiv:2401.17645*, 2024.

[33] Tianfeng Wu, Xiaofeng Liu, and Shoubin Dong. Ltrrs: a learning to rank based algorithm for resource selection in distributed information retrieval. In *Information Retrieval: 25th China Conference, CCIR 2019, Fuzhou, China, September 20–22, 2019, Proceedings 25*, pages 52–63. Springer, 2019.

[34] Guangzhi Xiong, Qiao Jin, Zhiyong Lu, and Aidong Zhang. Benchmarking retrieval-augmented generation for medicine. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics ACL 2024*, pages 6233–6251, Bangkok, Thailand and virtual meeting, August 2024. Association for Computational Linguistics.

[35] Dongfang Zhao. Frag: Toward federated vector database management for collaborative and secure retrieval-augmented generation. *arXiv preprint arXiv:2410.13272*, 2024.

[36] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Zhao, Andrew M Dai, Quoc V Le, James Laudon, et al. Mixture-of-experts with expert choice routing. *Advances in Neural Information Processing Systems*, 35:7103–7114, 2022.