ARTICLE | Article in Press

# Efficient training-free query routing framework for federated retrieval-augmented generation

**Hosun Eoo · Yeonho Yoo · Gyeongsik Yang**

Received: 28 May 2026 / Accepted: 9 October 2026
© The Author(s) 2026

**Abstract**

Retrieval-augmented generation (RAG) enhances large language models (LLMs) by grounding responses in external documents retrieved at inference time. In federated RAG (FedRAG), documents are distributed across multiple autonomous corpora that cannot be centrally aggregated due to privacy, regulatory, or organizational constraints. A key challenge in FedRAG is query routing: selecting a subset of relevant corpora for each query without broadcasting to all participants. Existing routing approaches rely on supervised learning, requiring a uniform retrieval embedding model across corpora and costly retraining whenever the federation changes. These assumptions violate corpus autonomy and limit scalability in dynamic federated environments. We propose RAGSpot , a training-free query routing framework that enables efficient corpus selection without requiring a shared local retrieval embedding model. RAGSpot represents each corpus using compact cluster-level metadata derived from its vector index and estimates corpus relevance via query-to-cluster matching. In addition, it introduces a rank-based fusion strategy that jointly leverages document-level and corpus-level rankings to support robust aggregation across heterogeneous embedding models. Experiments with corpora from the BEIR benchmark show that RAGSpot improves retrieval accuracy over the strongest training-based router by 1.23–**1.48**× in nDCG and 1.28–**1.66**× in Recall, and over the strongest LLM-based router by 1.63–**2.01**× in nDCG and 1.93–**2.34**× in Recall, without any further training. Also, RAGSpot achieves query latency that is comparable to, or even lower than, that of SOTA methods on average.

**Keywords** Large language models · Retrieval-augmented generation · Federated RAG · Query routing · Corpus selection

## Introduction

Large language models (LLMs) have been applied to various natural language processing tasks, including question answering[1], dialogue generation[2], and fact verification[3]. However, they may generate incorrect information or rely on outdated knowledge that does not reflect events that occur after pretraining, as in the case of hallucinations[4,5]. To address these limitations, retrieval-augmented generation (RAG) has been introduced, which retrieves

This Article in Press is shared early to give you faster access to new research. It is citable and carries a permanent DOI. The final edited version will replace it automatically.





https://doi.org/10.1038/s41598-026-75918-x

Published online: 13 October 2026

relevant documents from an external corpus at inference time and uses them to generate the final answer[6,7]. This enables the model to produce evidence-grounded, up-to-date responses beyond the data used for its training.

Recently, federated RAG (FedRAG) has attracted growing attention in LLM-based systems, where multiple distributed corpora collaborate to answer a query[8]. In this setting, each corpus is maintained independently at its own location, which offers advantages for privacy and data ownership. Each corpus performs local retrieval, and a central component aggregates the retrieved results to generate the final response.

Because documents are distributed, FedRAG must decide which corpora to query for each input. A straightforward approach is to broadcast the query to all available corpora and collect their results[9]. While this ensures full coverage, the communication cost scales with the number of corpora, making it inefficient in large-scale deployments. To reduce this cost, recent approaches propose selecting a subset of relevant corpora using trained routing mechanisms[10,11].

Despite its growing adoption, FedRAG still faces several practical challenges, as revealed by our analysis ("Challenges of training-based routing"). First, existing routing approaches assume a shared retrieval embedding model across all corpora, which violates corpus autonomy ("Violation of corpus autonomy"). Second, this assumption can degrade retrieval accuracy, as corpora often benefit from domain-specific embedding models tailored to their data ("Accuracy degradation from uniform embedding"). Third, supervised routers require costly retraining whenever the federation evolves, and this retraining depends on fresh relevance labels that are difficult to obtain in federated environments due to privacy constraints and annotation cost ("Retraining overhead").

To address these challenges, we propose RAGSpot, a training-free query routing framework for FedRAG. Our approach represents each corpus using cluster-level metadata, consisting of centroids and radii derived from its vector index, enabling efficient corpus-level relevance estimation. At query time, RAGSpot compares the query embedding with this metadata to route the query to the most relevant corpora. Since routing relies only on such metadata, each corpus can use its own retrieval encoder without requiring a shared retrieval embedding model; the compact routing metadata itself lies in a single lightweight routing space ("Metadata generation"). New corpora can be incorporated by simply registering their metadata, without retraining or access to raw documents. In addition, we design a rank-based fusion strategy that jointly considers document-level rankings and corpus-level relevance, avoiding direct comparison of heterogeneous similarity scores across embedding models.

We evaluate RAGSpot on a federated benchmark constructed from six diverse corpora from BEIR, and compare it against state-of-the-art (SOTA) baselines, including broadcast-based, training-based, and LLM-based methods ("Experiments"). The main contributions of this work are as follows:

- We propose a training-free query routing framework that operates solely on cluster-level metadata and, unlike existing FedRAG routers, jointly supports heterogeneous local retrieval encoders and a dynamically changing set of corpora without retraining.
- We introduce a rank-based fusion mechanism that jointly leverages document-level (local) and corpus-level (global) ranks, enabling robust aggregation across heterogeneous embedding models.
- RAGSpot consistently outperforms training-based, LLM-based, and broadcast baselines in retrieval accuracy, improving nDCG over the strongest training-based router by 1.23–1.48× and over the strongest LLM-based router by 1.63–2.01×.
- RAGSpot 's query latency is comparable to the broadcast baseline, 1.10–1.11× faster than the training-based router, and 4.5–6.9× faster than LLM-based routers.

## Background

Existing RAG techniques are categorized into two types: centralized and federated RAG. We explain the two types as the background of this study.

## Centralized RAG

Centralized RAG collects all external documents into a single corpus and performs retrieval over this unified dataset. This simplifies indexing and retrieval. We denote the single global corpus of documents as: $\mathcal{C} = \{d_1, \ldots, d_M\}$ , where each $d_i$ denotes a document segment used as a retrieval unit. Figure 1ashows the workflow.

**Indexing.** Each $d_i$ is encoded into a dense vector representation using an embedding model $\phi$ (❶ in Figure 1a). The resulting embeddings $\{\phi(d_1), \ldots, \phi(d_M)\} \subset \mathbb{R}^e$ are stored in an approximate nearest-neighbor (ANN) index. This index plays a role analogous to database indexing structures, but is designed for efficient similarity search in high-dimensional vector spaces, enabling scalable retrieval without exhaustive comparison over the entire corpus.

**Retrieval.** At query time, the query $q$ is encoded into its vector representation $\phi(q)$ using the same embedding model (❷). The retriever then performs a similarity search over the ANN index to identify $\mathcal{R}(q)$,the top-$k$ document segments (❸), whose embeddings are most similar to $\phi(q)$:

$$\mathcal{R}(q) = \underset{d_i \in \mathcal{C}}{\text{top-}k} \ \text{sim}(\phi(q), \phi(d_i)). \tag{1}$$

**Generation.** $\mathcal{R}(q)$ is combined with the original query $q$ to form an augmented prompt (❹), which serves as contextual evidence for generation. The LLM produces the final output $y$ by taking both the query and the retrieved documents as its inputs (❺):

$$y = \text{LLM}(q \,\|\, \mathcal{R}(q)), \tag{2}$$

where $\|$ denotes concatenation.

## Federated RAG

FedRAG extends the workflow of centralized RAG into a system of multiple corpora, $\mathcal{C} = \{C_1, \ldots, C_N\}$, which are managed independently[8]. Each corpus, $C_i$ maintains its own document collection and local retriever, and raw

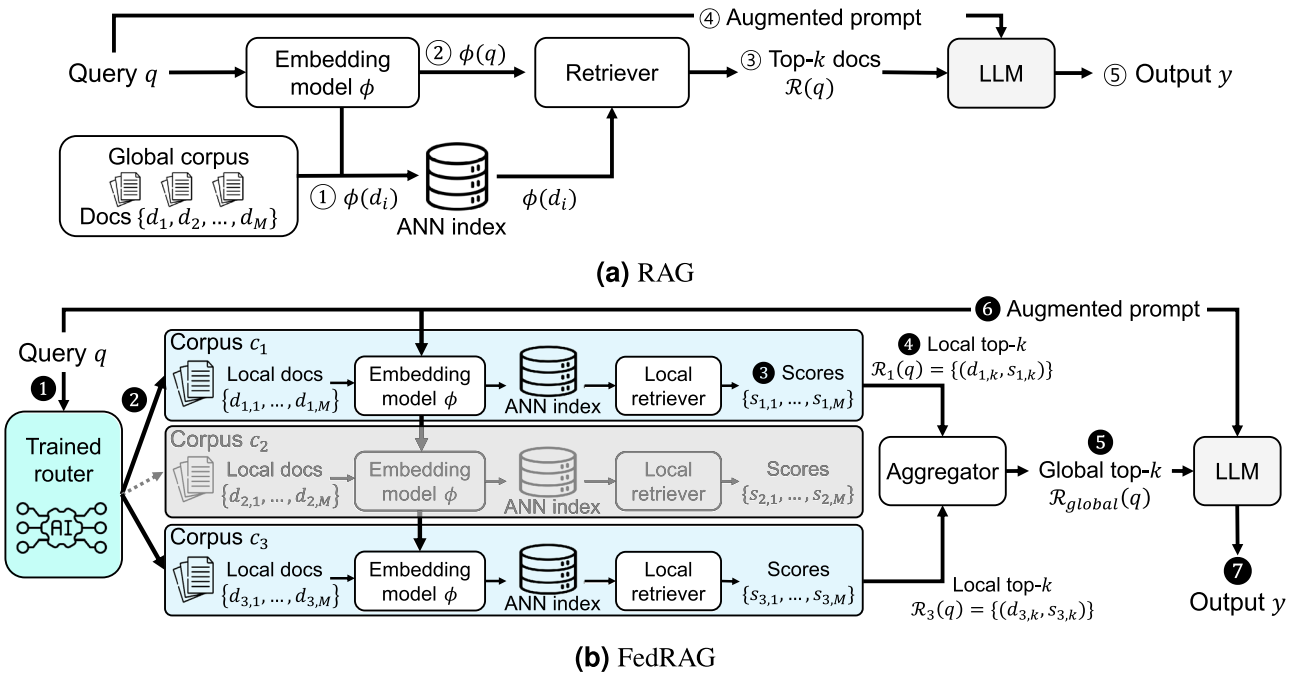



**Fig. 1** RAG and FedRAG architectures.

documents are not shared across corpora. Compared with centralized RAG, FedRAG introduces two additional components: (1) a corpus router, which selects relevant corpora for each query, and (2) an aggregator, which combines results from multiple corpora into a single ranked list. Figure 1bshows its architecture and workflow.

**Query routing.** Given a query $q$, the router selects a subset $S_q \subseteq \mathcal{C}$ of corpora to retrieve. Ideally, the optimal routing set $S_q^*$ is the smallest subset of corpora that contains all corpora with relevant documents for $q$. However, because the relevance of corpora to $q$ is unknown before conducting retrieval itself, existing techniques approximate this objective.

Several studies have proposed learning-based routing. For example, RAGRoute[10] uses a trained router (❶ in Figure 1b) to predict a subset of relevant corpora for a given query. Based on the prediction, the query is routed only to the selected corpora $S_q$ (❷). Each corpus then performs local retrieval over its own documents using its own ANN index, without sharing raw data. Here, each corpus maintains its own copy of the same embedding model $\phi$, ensuring that query and document representations across all corpora lie in a shared semantic space. This makes similarity scores directly comparable across corpora and supports reliable routing decisions. Each local retriever computes relevance scores (e.g., similarity between $\phi(q)$ and $\phi(d)$) for its retrieved documents (❸).

**Global Aggregation.** After routing, each selected corpus $C_i \in S_q$ returns its local top-$k$ results, $\mathcal{R}_i(q) = \{(d_{i,1}, s_{i,1}), \ldots, (d_{i,k}, s_{i,k})\}$ (❹), where each score $s_{i,j}$ denotes the relevance between the query and document $d_{i,j}$. The aggregator then merges candidates from all selected corpora and selects the global top-$K$ results (❺):

$$\mathcal{R}_{\text{global}}(q) = \underset{d \in \bigcup_{C_i \in S_q} \mathcal{R}_i(q)}{\text{top-}K} \text{sim}(\phi(q), \phi(d)). \tag{3}$$

The selected global top-$K$ documents $\mathcal{R}_{\text{global}}(q)$ are concatenated with the query $q$ to form an augmented prompt (❻), which is then fed into the LLM to generate the final output $y$ (❼).

## Challenges of training-based routing

In this section, we demonstrate the practical challenges of the existing FedRAG. We focus on learning-based routing, which is the state-of-the-art.

### Violation of corpus autonomy

A core principle of FedRAG is that each corpus independently manages its own indexing pipeline: documents are encoded into vector representations and stored in a local vector index for similarity search. Because each corpus operates autonomously, it can choose its own embedding model and indexing strategy; in practice, different corpora often adopt different models tailored to their domain characteristics.

Despite this principle, existing FedRAG techniques implicitly assume a shared embedding model across all corpora. For example, RAGRoute uses a unified embedding space in which both queries and corpus representations are encoded using the same model. Its router relies on embedding-based features, such as query embeddings, corpus centroids, and distances between them, which are meaningful only when all vectors lie in the same space with consistent dimensionality. This assumption directly conflicts with corpus autonomy: in practice, different organizations select embedding models optimized for their specific domains (e.g., biomedical[12] or legal corpora[13]), and requiring a single shared model undermines this independence while introducing coordination overhead that grows with the number of corpora.

### Accuracy degradation from uniform embedding

Beyond autonomy, enforcing a shared embedding model can also degrade retrieval effectiveness. Since corpora often exhibit different domain characteristics and document distributions, the encoder that performs best for one

corpus may not be optimal for another[14,15]. Restricting all corpora to a single encoder thus prevents them from leveraging models that best capture their local semantics.

To quantify this effect, we compare two settings: (1) uniform setting where all corpora share a single encoder, as existing techniques do, and (2) corpus-specific setting where each corpus uses its own optimal encoder.

**Experiment setting.** The uniform setting uses `UAE-Large-V1`[16]across all corpora. For the corpus-specific setting, we adopt the best-performing encoder reported in Wang et al[15]. for the three datasets included in Table 1: SciFact (`gte-base`[17]), SciDocs (`all-mpnet-base-v2`[18]), and FiQA (`all-mpnet-base-v2`[18]). All experiments share the same retrieval pipeline and index configuration. We report nDCG and Recall at cutoffs $k \in \{1, 10\}$.

All experiments are conducted on a server equipped with an Intel Xeon CPU, two NVIDIA A100 (32 GB) GPUs, two NVIDIA RTX 3090 (24 GB) GPUs, and 1,024 GB of RAM, running Ubuntu 22.04. All results are averaged over three runs.

**Results.** Table 1 summarizes the results. Across the three datasets, the corpus-specific encoder consistently outperforms the shared encoder. The gap is most pronounced on FIQA, where nDCG@10 and Recall@10 drop by 5.41% and 6.55%, respectively, under the uniform setting. SciFact and SciDocs show smaller but consistent degradations of 2–3% across metrics. The results indicate that the optimal embedding model varies across corpora and that enforcing a uniform encoder systematically hurts retrieval quality.

## Retraining overhead

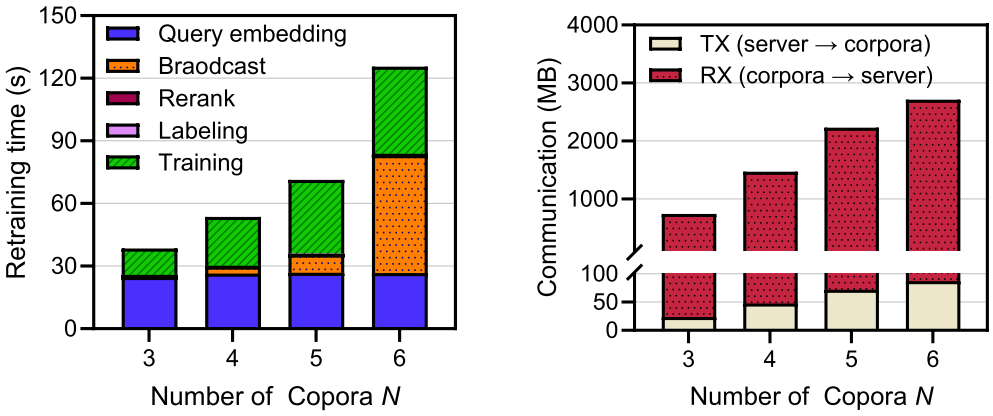
Another key limitation is the cost of retraining. Existing techniques rely on supervised models whose parameters encode the relationship between query representations and corpus-level features. Whenever the corpus changes, such as when a corpus joins, leaves, or updates its data, the learned parameters become stale, and the router must be retrained before it can route queries reliably. This retraining also requires fresh relevance labels, which are difficult to obtain in FedRAG settings, as sharing query logs or relevance annotations may violate data-governance constraints, and exhaustive labeling across multiple corpora is expensive.

During this retraining process, the system cannot serve accurate routing decisions, making retraining latency a direct bottleneck on RAG availability. Figure 2aconfirms this concern: as the number of corpora grows from three to six, total retraining time rises from 38.4 s to 125.5 s, during which the system is unable to serve queries.

Retraining is also communication-intensive. Constructing an updated training set requires a full replay of the retrieval pipeline: for each of the $Q$ training queries, the system broadcasts the query to all $N$ corpora, collects top-$k$ results from each, and merges them into a global ranking (❺ in Figure 1b) to derive supervision signals. Crucially, even unchanged corpora must be re-queried because the global ranking depends on the full set of participants. This amounts to $N \times Q$ retrieval requests and the transfer of $N \times Q \times k$ results; in our experiment (Figure

**Table 1** Retrieval performance comparison between a uniform embedding model and corpus-specific optimal models. Negative values indicate performance degradation under the uniform embedding setting.

| Metric | Setting | SciFact | SciDocs | FiQA |
|---|---|---|---|---|
| nDCG@1 | Uniform | 0.63 | 0.24 | 0.44 |
| | Optimal | 0.65 | 0.26 | 0.49 |
| | **Difference (%)** | **−2.00** | **−1.40** | **−5.09** |
| nDCG@10 | Uniform | 0.74 | 0.22 | 0.45 |
| | Optimal | 0.76 | 0.24 | 0.50 |
| | **Difference (%)** | **−2.27** | **−1.72** | **−5.41** |
| Recall@1 | Uniform | 0.60 | 0.05 | 0.22 |
| | Optimal | 0.62 | 0.05 | 0.24 |
| | **Difference (%)** | **−1.83** | **−0.31** | **−2.84** |
| Recall@10 | Uniform | 0.86 | 0.24 | 0.52 |
| | Optimal | 0.89 | 0.26 | 0.58 |
| | **Difference (%)** | **−3.26** | **−2.04** | **−6.55** |




**(a)** Time overhead. **(b)** Communication overhead.

**Fig. 2** Retraining overhead of training-based routing as the number of corpora $N$ increases. (a) Time overhead is broken down by pipeline stage; the broadcast stage grows fastest and dominates at $N$=6. (b) Communication overhead quantifies the data exchanged between the central server and corpora by TX (server → corpora) and RX (corpora → server).

2b), communication volume increases from 737.8 MB to 2709.7 MB as the number of corpora grows from three to six.

### Summary

Existing learning-based routing approaches fundamentally assume a homogeneous, static federation. They require a uniform embedding model, thereby violating corpus autonomy ("Violation of corpus autonomy") and degrading retrieval accuracy when corpora benefit from domain-specific encoders ("Accuracy degradation from uniform embedding"). They also require costly retraining whenever the federation changes, blocking RAG availability and incurring communication overhead that scales with the number of corpora ("Retraining overhead"). These limitations motivate a new routing approach that operates without a shared retrieval encoder or model retraining.

## RAGSpot design

To address the three challenges identified in "Challenges of training-based routing", we design RAGSpot around three corresponding components. Metadata generation ("Metadata generation") addresses federation dynamics by allowing a corpus to join or update the system by publishing compact routing metadata instead of retraining a router. Corpus filtering ("Corpus filtering") addresses autonomy and uniform-encoder degradation by scoring corpora through radius-normalized cluster metadata rather than shared retrieval embeddings. Rank-based fusion ("Rank-based fusion") addresses score incomparability across heterogeneous local retrievers by combining corpus-level and document-level ranks instead of raw similarity scores. Figure 3 illustrates the overall architecture and workflow of RAGSpot.

First, prior to query processing, each corpus constructs statistical representations of its semantic distribution. Whenever a corpus is added or updated, the metadata is updated in the metadata registry (① in Figure 3). This stage is performed independently of query processing. Next, given a query, RAGSpot encodes it into a query vector (②), and the RAGSpot router selects a subset of relevant corpora based on the query vector and the corpus metadata (③). Each selected corpus then returns its top-$k$ documents for the query to the RAGSpot aggregator (④). The RAGSpot aggregator derives a single merged ranked list of documents from the selected corpora by considering both local document rankings and global corpus-level rankings (⑤). Finally, the merged documents (⑥) and augmented prompt (⑦) are fed into the LLM to generate the final output $y$ (⑧). We detail each stage in the following subsections.

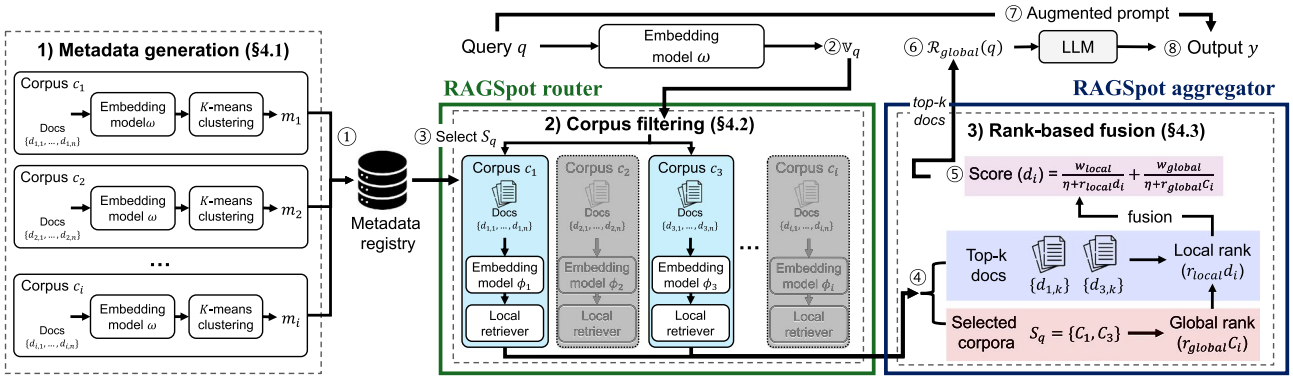



**Fig. 3** Architecture of RAGSpot.

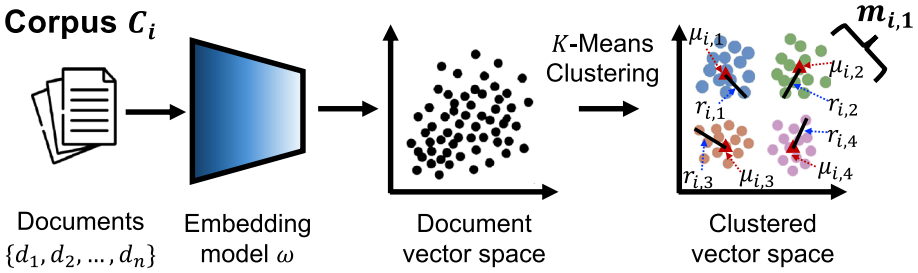



**Fig. 4** The metadata $(m_i)$ generation process for corpus $C_i$. Raw documents $\{d_{i,1}, d_{i,2}, \ldots, d_{i,n}\}$ are encoded into a vector space using an embedding model. The document vectors are then partitioned using $K$-Means clustering. For each cluster $(j)$, a centroid $(\mu_{i,j})$ and a radius $(r_{i,j})$ are computed to form the routing metadata.

## Metadata generation

The metadata generation stage constructs statistical representations of the semantic distribution for each corpus, which is performed independently within each corpus and prior to query processing. Only the resulting metadata is shared with the RAGSpot router—no raw documents are exposed to other corpora.

As illustrated in Figure 3, $C_i$ generates and maintains its metadata $m_i$ whenever the corpus is newly added or updated. Figure 4 shows the detailed metadata generation process. First, all documents in $C_i$ are encoded into the shared routing space using the common routing encoder $\omega$ (e.g., UAE-Large-V1). The routing encoder $\omega$ is used only to generate routing metadata and to encode routing queries; it is distinct from the corpus-specific local retrieval encoder $\phi_i$ used for local retrieval after routing. Second, the resulting document vectors are partitioned into $K$ clusters using the $K$-Means algorithm. For each cluster $j \in \{1, \ldots, K\}$, the following values are calculated and recorded:

- Centroid $\mu_{i,j}$: the mean vector of all document embeddings of cluster $j$.
- Radius $r_{i,j}$: the 95th-percentile distance from the centroid to documents in cluster $j$, which captures cluster extent while remaining less sensitive to outliers than the maximum distance.

Thus, $m_i$ for $C_i$ is defined as:

$$m_i = \{(\mu_{i,j},\ r_{i,j}) \mid j = 1, \ldots, K\}. \tag{4}$$

Each corpus transmits only $m_i$ to the RAGSpot, which stores it in the metadata registry, ensuring that raw documents remain private. The communication cost per corpus is $O(K \cdot e)$, where $e$ is the embedding dimensionality, which is negligible compared to transmitting document-level embeddings.

## Corpus filtering

As explained in "Violation of corpus autonomy", existing techniques assume an identical embedding model across $N$ corpora, mapping both queries and corpus representations into a unified embedding space, and select corpora by comparing query embeddings with corpus-level representations using a predefined global threshold. However, this conflicts with corpus autonomy, where each corpus would employ its own domain-specific embedding model. Even when a unified space is enforced, heterogeneous semantic distributions across corpora make a single global threshold either overly restrictive or overly permissive.

To address the limitation, RAGSpot avoids relying on the same embedding model, and instead, leverages metadata $m_i = \{(\mu_{i,j}, r_{i,j})\}$, which provides a cluster-level representation of each corpus without exposing document-level embeddings. Specifically, given a query $q$, RAGSpot encodes it into a routing query vector $\mathrm{v}_q = \omega(q)$ using the same shared routing encoder $\omega$ that generated the metadata (② in Figure 3) and estimates corpus relevance by jointly considering the positions of clusters (i.e., centroids $\mu_{i,j}$) and their extents (i.e., radii $r_{i,j}$). Because all centroids $\mu_{i,j}$ and $\mathrm{v}_q$ are produced by $\omega$, every query-to-centroid distance is computed within a single routing space and is directly comparable across corpora.

Unlike prior approaches that require identical retrieval encoders, RAGSpot decouples local retrieval from routing: each corpus $C_i$ may use its own retrieval model $\phi_i$, used only for local retrieval after routing, while routing operates on shared centroid–radius metadata generated by the common encoder $\omega$ in a single routing space.

**Routing space vs. retrieval space.** RAGSpot decouples two representation spaces. The *routing space* is shared: one lightweight encoder $\omega$ maps all routing metadata and routing queries into a common space, which makes cross-corpus distance comparison well defined. The *retrieval space* is corpus-local: each $C_i$ keeps its own $\phi_i$. Radius normalization only rescales distances *within* this shared routing space; it does not attempt to align heterogeneous spaces. Furthermore, by leveraging cluster-level information rather than relying on a single global threshold, RAGSpot enables adaptive corpus scoring that is robust to variations in semantic scale and density across corpora. Further details are provided in the following subsections.

### Corpus scoring

**Centroid-based cluster search.** For each corpus $C_i$, the RAGSpot router identifies the cluster whose centroid is closest to $\mathrm{v}_q$. Let $\{\mu_{i,1}, \ldots, \mu_{i,K}\}$ denote the centroids of $C_i$. The nearest cluster index is computed as:

$$j_i^* = \arg\min_{j \in \{1,\ldots,K\}} \|\mathrm{v}_q - \mu_{i,j}\|_2. \tag{5}$$

This step identifies the region within each corpus that is most relevant to the query. By operating on cluster centroids instead of individual document embeddings, the computational cost is reduced from $O(|d_i|)$ to $O(K)$ per corpus.

**Radius-based score normalization.** The distance to the nearest centroid alone is insufficient for comparing corpora, as each corpus has a different embedding distribution with varying spread. So, the same distance may indicate high relevance in a tightly clustered corpus but low relevance in a widely spread one, making raw distances incomparable across corpora. For example, consider two corpora (A and B) with cluster radius values of 1.0 and 0.2, respectively, representing their characteristic distribution scales. For a given query $q$, assume that the distance to the nearest centroid is identical (0.5) in both corpora. For corpus A, this distance is smaller than its cluster radius ($1.0 > 0.5$), indicating that the query lies within the cluster and is well aligned with the corpus distribution. In contrast, for corpus B, the same distance exceeds its cluster radius ($0.2 < 0.5$), indicating that the query lies outside the cluster and is less aligned with the corpus distribution.

To enable consistent comparison, we normalize the query-to-centroid distance using the radius of the corresponding cluster:

$$S_i = \frac{\|\mathrm{v}_q - \boldsymbol{\mu}_{i,j_i^*}\|_2}{r_{i,j_i^*}}. \tag{6}$$

The normalized score $S_i$ represents the distance between the query and the nearest cluster center relative to the cluster scale. A smaller $S_i$ indicates that the query is closer to the core region of the corpus, while a larger value suggests lower relevance. This normalization allows scores to be comparable across corpora with different scales and densities. Because $S_i$ is a ratio of two quantities expressed in the same units, it is dimensionless. The selection threshold $\gamma$ introduced in "Corpus selection" therefore carries the same meaning for every corpus, regardless of how tightly or loosely that corpus is distributed, and a single global setting of $\gamma$ remains well defined as corpora are added or replaced. Normalization is in this sense a calibration step that makes the threshold transferable across corpora, rather than a mechanism that sharpens relevance estimation within any individual corpus.

### Corpus selection

Based on the normalized scores, RAGSpot selects a subset of corpora whose scores are close to the most relevant corpus. Let $S_{\min}$ denote the best (i.e., smallest) score across all corpora. We define a filtering threshold as:

$$\tau = S_{\min} \cdot \gamma, \tag{7}$$

where $\gamma \geq 1$ is a gap ratio hyperparameter that controls the allowable deviation from the best score, thereby determining the selectivity of the filter; $\gamma = 1$ selects only the best-scoring corpus, and values below 1 are invalid as they would yield an empty selection.

Finally, the selected corpus set is:

$$S_q = \{C_i \mid S_i \leq \tau\}. \tag{8}$$

A larger $\gamma$ includes more corpora by relaxing the threshold, while a smaller $\gamma$ results in more selective filtering. We set $\gamma$ to 1.3 in our experiments; the sensitivity of this parameter is analyzed in "Hyperparameter study".

## Rank-based fusion

After corpus selection, each selected corpus $C_i \in S_q$ performs local retrieval using its own retriever and embedding model $\phi_i$, and returns its top-$k$ documents to the RAGSpot aggregator (④ in Figure 3). This local retrieval step follows the standard FedRAG setting, where each corpus independently retrieves documents from its own data without sharing raw documents. The RAGSpot aggregator then combines the retrieved documents from multiple corpora into a single ranked list (⑤). A key challenge in this step is that different corpora may employ different embedding models and scoring values, making their similarity scores not directly comparable across corpora. As a result, aggregating documents based on raw scores can lead to incorrect global rankings.

For example, consider two corpora $C_1$ and $C_2$ selected for a query $q$, where $C_1$ is more relevant to the query at the corpus level than $C_2$. Suppose $C_1$ returns a document ranked 5th in its local results, while $C_2$ returns a document ranked 1st. If we rely only on local document ranks, the document from $C_2$ would be ranked higher, even though it originates from a less relevant corpus. On the other hand, if we rely only on corpus-level relevance, all documents from $C_1$ would be ranked above those from $C_2$, ignoring useful distinctions between documents within each corpus. This illustrates that neither local document rank nor corpus-level relevance alone is sufficient. Effective aggregation requires jointly considering both document-level and corpus-level signals.

To address this issue, we adopt a rank-based fusion strategy that relies on both document-level and corpus-level ranks. For each document $d_{i,l}$ returned by corpus $C_i$, we define two rank components:

- Local document rank $r_{\text{local}}(d_{i,l})$: the rank of $d_{i,l}$ within the top-$k$ list returned by $C_i$, reflecting document-level relevance within the corpus.

- Global corpus rank $r_{\text{global}}(C_i)$: the rank of corpus $C_i$ among the selected corpora, obtained by sorting the normalized scores $\{S_i \mid C_i \in S_q\}$ in ascending order.

The fused score for a document $d_{i,l}$ from corpus $C_i$ is computed as:

$$\text{Score}(d_{i,l}) = \frac{w_{\text{local}}}{\eta + r_{\text{local}}(d_{i,l})} + \frac{w_{\text{global}}}{\eta + r_{\text{global}}(C_i)}, \tag{9}$$

where $\eta$ is a smoothing constant corresponding to the reciprocal rank fusion (RRF) constant, which controls the influence of rank positions by reducing the dominance of top-ranked results. The weights $w_{\text{local}}$ and $w_{\text{global}}$ (fusion weights) determine the relative importance of document-level and corpus-level signals. We set $\eta = 60$ and use fusion weights with a ratio of $w_{\text{local}} : w_{\text{global}} = 1 : 5$, prioritizing corpus-level relevance over local document ranking (sensitivity analysis in "Hyperparameter study"). This reflects the intuition that documents from more relevant corpora are likely to be relevant even if their local ranks are lower.

All retrieved documents from $C_i \in S_q$ are scored using Eq. (9) and sorted in descending order. The top-$K$ documents are selected as the final retrieval result $\mathcal{R}_{\text{global}}(q)$ (⑥). Finally, with the augmented query (⑦), the LLM generates the final output $y$ (⑧).

Algorithm 1 presents the complete query-processing pipeline of RAGSpot.

**Require:** $q$; $\{m_i\}_{i=1}^{N}$; $\omega$; $\gamma$; $\eta$; $w_{\text{local}}, w_{\text{global}}$; $K$
**Ensure:** $\mathcal{R}_{\text{global}}(q)$
$\mathbf{v}_q \leftarrow \omega(q)$
**for** each corpus $C_i$ with $m_i = \{(\mu_{i,j}, r_{i,j})\}_{j=1}^{K}$ **do**
  $j_i^* \leftarrow \arg\min_j \|\mathbf{v}_q - \mu_{i,j}\|_2$
  $S_i \leftarrow \|\mathbf{v}_q - \mu_{i,j_i^*}\|_2 / r_{i,j_i^*}$
**end for**
$\tau \leftarrow \gamma \cdot \min_i S_i$
$S_q \leftarrow \{C_i \mid S_i \le \tau\}$
**for** each $C_i \in S_q$ **in parallel do**
  $\mathcal{R}_i(q) \leftarrow \text{Retrieve}_i(\phi_i(q))$
  Rank $d_{i,l}$ in $\mathcal{R}_i(q) \rightarrow r_{\text{local}}(d_{i,l})$
**end for**
Rank $C_i \in S_q$ by $S_i$ (ascending) $\rightarrow r_{\text{global}}(C_i)$
**for** each $d_{i,l} \in \mathcal{R}_i(q)$, $C_i \in S_q$ **do**
  $\text{Score}(d_{i,l}) \leftarrow \frac{w_{\text{local}}}{\eta + r_{\text{local}}(d_{i,l})} + \frac{w_{\text{global}}}{\eta + r_{\text{global}}(C_i)}$
**end for**
$\mathcal{R}_{\text{global}}(q) \leftarrow \text{TopK}(\text{Score}, K)$
**return** $\mathcal{R}_{\text{global}}(q)$

**Algorithm 1** RAGSpot query processing

## Properties and complexity

RAGSpot requires all routing centroids and the query representation $\mathrm{v}_q$ to lie in the shared routing space defined by $\omega$; this is a precondition rather than a theoretical result ("Corpus filtering").

Several properties follow directly from the selection rule. First, because the normalized scores satisfy $S_i \ge 0$, the minimum-score corpus always satisfies $S_i \le S_{\min} \cdot \gamma$ for $\gamma \ge 1$, and thus $S_q$ is non-empty; conversely, for $\gamma < 1$ no corpus can satisfy the condition and routing collapses, which is why $\gamma \ge 1$ is required by construction. Second, $S_q$ is monotonically non-decreasing with $\gamma$. Consequently, corpus-level routing recall cannot decrease as $\gamma$ increases, making $\gamma$ an explicit control over the routing recall–cost trade-off. If $S_{\min} > 0$, the rule selects all corpora once $\gamma \ge \max_i S_i / S_{\min}$, at which point it becomes equivalent to broadcasting. The normalized score is also invariant when all query-to-centroid distances and cluster radii in the routing space are multiplied by the same positive constant.

Computing the routing scores requires $O(NKd)$ time per query, where $N$ is the number of corpora, $K$ is the number of clusters per corpus, and $d$ is the routing-space dimension; this reduces to $O(NK)$ when $d$ is treated as fixed. For $M = |S_q|k$ retrieved candidates, rank-score accumulation requires $O(M)$ time and sorting the candidates requires $O(M \log M)$ time. Neither cost depends on the number of documents each corpus holds ("Query overhead").

These are structural properties of the routing and fusion rules. We make no distribution-free guarantee that RAGSpot recovers the oracle corpus set $S_q^*$ or approximates optimal retrieval quality; establishing such bounds would require distributional assumptions on how relevant documents are placed across corpora, which we regard as outside the scope of this work.

## Experiments

We evaluate RAGSpot in a FedRAG setting constructed from multiple heterogeneous corpora. Our goal is to evaluate (1) routing accuracy on both composite and single-corpus queries, (2) end-to-end generation quality, (3) query overhead, and (4) the overhead of metadata generation of RAGSpot. We additionally study the sensitivity of RAGSpot to its key hyperparameters.

### Experiment setting

**Environment.** All experiments are conducted under the same environment as in "Challenges of training-based routing". We construct a FedRAG environment using six datasets from the BEIR benchmark, treating each dataset as an independent corpus, $\mathcal{C} = \{C_1, \ldots, C_6\}$. The corpora include SciFact[19], ArguAna[20], FiQA[21], NFCorpus[22], SciDocs[23], and Webis-Touche-2020[24]. Table 2 summarizes the number of queries and documents in each corpus.

**Query workloads.** We use two types of query workloads, single-corpus queries and composite queries.

- **Single-corpus queries.** Each query is drawn from a single BEIR corpus, resulting in 3,726 samples. A document is considered relevant based on the BEIR ground-truth annotations, and correct routing requires selecting the corresponding corpus.
- **Composite (multi-corpus) queries.** To evaluate performance under multi-domain information needs, we construct composite queries that span two corpora. For each pair of distinct corpora $(C_A, C_B)$, we generate a composite query by concatenating test queries $q_A$ and $q_B$ (i.e., $q_{\text{comp}} = [q_A; q_B]$). We consider all 15 possible two-corpus combinations among the six corpora, resulting in 3,464 samples. A document is considered relevant if it is relevant to either $q_A$ or $q_B$ according to the BEIR ground-truth annotations; thus, correct routing requires selecting *both* $C_A$ and $C_B$. We treat this workload as a controlled synthetic stress test: concatenation yields exact two-source ground truth, which isolates routing behavior, but it does not by itself represent naturally phrased multi-domain requests. We therefore also construct a naturally phrased variant of the same workload, described in "Naturally phrased composite queries".

**Hyperparameters of** RAGSpot **.** We select $\gamma$, $\eta$, and $w_{\text{local}} : w_{\text{global}}$ based on the sensitivity analysis in "Hyperparameter study". Specifically, $\gamma = 1.3$ balances retrieval effectiveness against the number of selected corpora, while $\eta = 60$ and $w_{\text{local}} : w_{\text{global}} = 1:5$ provide stable performance across the evaluated retrieval metrics. The number of $k$-means clusters per corpus is set proportional to the corpus size. Each corpus uses FAISS[25] with an IVF-Flat index for ANN retrieval, with a local retrieval depth of $k = 30$ documents per corpus. The cluster radius is defined using the $P_{95}$ percentile to reduce sensitivity to outliers.

**Baselines.** We compare RAGSpot against the following SOTA baselines:

**Table 2** Detailed statistics of the BEIR corpora evaluated in our experiments.

| Corpus | Queries | Documents |
|---|---|---|
| NFCorpus | 323 | 3,633 |
| SciFact | 300 | 5,183 |
| ArguAna | 1,406 | 8,674 |
| SciDocs | 1,000 | 25,657 |
| FiQA | 648 | 57,638 |
| Webis-Touche-2020 | 49 | 382,545 |

- **C-FedRAG**[9]: A broadcast-based FedRAG system where the router selects all corpora for each query. Each corpus performs local retrieval independently and returns its top-$k$ results, which are merged by the aggregator to produce the final global top-$K$ results for generation.
- **RAGRoute**[10]: A training-based FedRAG system where the router uses a multi-layer perceptron (MLP) to predict a subset of relevant corpora for each query based on features derived from query and corpus representations. Only the selected corpora perform local retrieval, and their results are aggregated.
- **ReSLLM**[11]: An LLM-based FedRAG system that uses a large language model as the router to select relevant corpora via prompting, i.e., directly asking the LLM which corpora are most relevant to the query. We use two variants, ReSLLM-flan and ReSLLM-SOLAR, which use FLAN-T5-XL[26] and SOLAR-11B[27] as backbone LLMs, respectively.**Statistical testing.** We report paired bootstrap 95% confidence intervals over the query set (10, 000 resamples) together with a two-sided sign test on per-query wins and losses, comparing RAGSPOT against each baseline on the same queries.

**Comparison protocol.** To ensure a fair comparison, all methods are evaluated using the same test queries, local retrieval depth of $k = 30$ documents per selected corpus, evaluation metrics and cutoffs, and hardware. The local retrieval procedure is identical for all methods; aggregation is part of each method's own design. C-FedRAG retrieves from every corpus and therefore examines the largest candidate pool, ensuring that it is not disadvantaged by a limited retrieval budget. Each baseline otherwise follows its original design. In particular, training-based baselines (e.g., RAGRoute) assume a shared embedding model across corpora, whereas RAGSPOT supports independently selected local encoders, which is the capability evaluated in "Accuracy degradation from uniform embedding".

**Evaluation metrics.** We evaluate RAGSPOT along four dimensions: (1) retrieval accuracy, (2) end-to-end generation quality, (3) system overhead (query latency and metadata generation), and (4) hyperparameter sensitivity. We use the following metrics.

- **Retrieval accuracy.** We report nDCG@$k$ and Recall@$k$ for $k \in \{1, 5, 10, 30\}$ on both the composite-query and single-query workloads. nDCG@$k$ evaluates ranking quality by assigning higher importance to relevant documents appearing at higher ranks, while Recall@$k$ measures how many relevant documents are covered within the top-$k$ results, regardless of their positions.
- **Generation quality.** On the end-to-end generation experiment ("End-to-end generation quality"), we report two reference-free RAGAS metrics: answer relevancy, which measures how well the generated answer addresses the query, and context utilization, which measures how effectively the retrieved context supports the answer.
- **Query overhead.** The per-query online cost, comprising the average end-to-end wall-clock latency and the communication volume exchanged between the router and the selected corpora ("Query overhead").
- **Metadata generation overhead.** The one-time wall-clock time required for each corpus to compute its local clusters and generate centroid–radius metadata used by RAGSPOT.

## Retrieval accuracy

Table 3 reports retrieval accuracy on the composite-query workload, where a correct router must select multiple relevant corpora for each query. To provide a more robust comparison, we report the average improvement across cutoffs $k \in 1, 5, 10, 30$. Compared to RAGRoute, RAGSPOT improves nDCG and Recall by 1.48× (up to 1.64×) and 1.66× (up to 1.83×) on average, demonstrating consistent gains even without training. Interestingly, C-FedRAG, despite querying all corpora (broadcasting), performs substantially worse. In particular, RAGSPOT achieves 2.33× (up to 3.00×) higher nDCG and 2.72× (up to 3.67×) higher Recall on average. This behavior is consistent with C-FedRAG's score-based aggregation. Because the corpora use different local retrieval encoders, their

**Table 3** Retrieval accuracy on the composite-query workload. Best results are in **bold**, and second-best results are underlined.

| Method | nDCG@1 | nDCG@5 | nDCG@10 | nDCG@30 | Recall@1 | Recall@5 | Recall@10 | Recall@30 |
|---|---|---|---|---|---|---|---|---|
| C-FedRAG | 0.13 | 0.07 | 0.06 | 0.06 | 0.03 | 0.04 | 0.05 | 0.06 |
| RAGRoute | 0.13 | 0.11 | 0.11 | 0.11 | 0.02 | 0.07 | 0.10 | 0.12 |
| ReSLLM-flan | 0.09 | 0.06 | 0.05 | 0.05 | 0.02 | 0.03 | 0.04 | 0.06 |
| ReSLLM-SOLAR | 0.16 | 0.10 | 0.09 | 0.09 | **0.03** | 0.06 | 0.07 | 0.09 |
| RAGSpot (ours) | **0.18** | **0.16** | **0.16** | **0.18** | **0.03** | **0.12** | **0.16** | **0.22** |

**Table 4** Retrieval accuracy on the single-query workload. Best results are in **bold**, and second-best results are underlined.

| Method | nDCG@1 | nDCG@5 | nDCG@10 | nDCG@30 | Recall@1 | Recall@5 | Recall@10 | Recall@30 |
|---|---|---|---|---|---|---|---|---|
| C-FedRAG | 0.07 | 0.07 | 0.08 | 0.08 | 0.05 | 0.08 | 0.09 | 0.11 |
| RAGRoute | 0.18 | 0.26 | 0.28 | 0.31 | 0.08 | 0.32 | 0.39 | 0.48 |
| ReSLLM-flan | 0.07 | 0.08 | 0.08 | 0.09 | 0.05 | 0.08 | 0.10 | 0.12 |
| ReSLLM-SOLAR | 0.15 | 0.15 | 0.16 | 0.17 | 0.09 | 0.15 | 0.18 | 0.23 |
| RAGSpot (ours) | **0.22** | **0.32** | **0.35** | **0.38** | **0.10** | **0.40** | **0.51** | **0.63** |

**Table 5** Statistical significance: Paired differences (RAGSpot minus baseline) with bootstrap 95% confidence intervals.

| Baseline | Workload | nDCG@10 | Recall@10 |
|---|---|---|---|
| RAGRoute | single | +0.038 [+0.027, +0.049] | +0.069 [+0.052, +0.085] |
| | composite | +0.038 [+0.033, +0.044] | +0.049 [+0.043, +0.054] |
| ReSLLM-flan | single | +0.262 [+0.249, +0.275] | +0.404 [+0.385, +0.423] |
| | composite | +0.104 [+0.098, +0.110] | +0.116 [+0.110, +0.122] |
| ReSLLM-SOLAR | single | +0.184 [+0.172, +0.196] | +0.322 [+0.302, +0.341] |
| | composite | +0.067 [+0.061, +0.072] | +0.091 [+0.085, +0.097] |
| C-FedRAG | single | +0.269 [+0.256, +0.282] | +0.417 [+0.398, +0.436] |
| | composite | +0.095 [+0.090, +0.100] | +0.111 [+0.106, +0.117] |

raw similarity scores are not directly calibrated across corpora, which can distort the global ranking. RAGSpot instead fuses corpus- and document-level ranks, avoiding direct comparisons between heterogeneous similarity scores.

Among LLM-based routers, ReSLLM-SOLAR outperforms ReSLLM-flan, achieving 1.54× (up to 1.67× ) higher nDCG and 1.66× (up to 1.75×) higher Recall on average, suggesting that stronger backbone LLMs improve routing quality. However, RAGSpot still achieves 1.63× (up to 2.00×) higher nDCG and 1.93× (up to 2.44×) higher Recall compared to ReSLLM-SOLAR, indicating that prompting-based routing lacks the fine-grained discrimination needed for accurate corpus selection.

Table 4 reports retrieval accuracy on the single-query workload. Compared to RAGRoute, RAGSpot improves nDCG by 1.23× (up to 1.25×) on average and Recall by 1.28× (up to 1.31×) on average. C-FedRAG remains significantly less effective, with RAGSpot achieving 4.21× (up to 4.75×) higher nDCG and 4.60× (up to 5.73×) higher Recall on average. Compared to LLM-based routers, RAGSpot outperforms ReSLLM-SOLAR by 2.01× (up to 2.24×) in nDCG and 2.34× (up to 2.83×) in Recall on average.

**Statistical significance.** Table 5 reports the paired difference between RAGSpot and each baseline (RAGSpot minus baseline) with bootstrap 95% confidence intervals (10, 000 resamples), for both nDCG@10 and Recall@10 on the single- and composite-query workloads. The paired differences favor RAGSpot in every comparison, with the narrowest margin over RAGRoute, the strongest baseline (+0.038 in single-corpus nDCG@10); the margins over C-FedRAG and the prompting-based routers are substantially larger. The confidence intervals exclude zero, and the two-sided sign test gives $p < 0.001$ for all comparisons except RAGRoute on single-corpus nDCG@10, where $p = 0.006$.

**Table 6** RAGSpot design ablation.

| Variant | nDCG@10 | Recall@10 | nDCG@30 | Recall@30 | Selected corpora |
|---|---|---|---|---|---|
| Full RAGSpot | 0.345 | 0.506 | 0.379 | 0.630 | 4.54 |
| w/o corpus filtering | 0.346 (+0.5%) | 0.507 (+0.1%) | 0.381 (+0.6%) | 0.632 (+0.5%) | 6.00 |
| w/o rank-based fusion | 0.079 (−76.9%) | 0.096 (−81.0%) | 0.084 (−77.7%) | 0.114 (−81.9%) | 4.54 |

**Table 7** Rank-based fusion design ablation.

| Variant | nDCG@10 | Recall@10 | nDCG@30 | Recall@30 |
|---|---|---|---|---|
| Full RAGSpot (corpus : doc = 1:5) | 0.345 | 0.506 | 0.379 | 0.630 |
| Corpus-rank-only | 0.338 (−1.8%) | 0.476 (−5.9%) | 0.365 (−3.5%) | 0.564 (−10.5%) |
| Local-rank-only (standard RRF) | 0.198 (−42.6%) | 0.385 (−23.9%) | 0.243 (−35.9%) | 0.545 (−13.5%) |
| Equal weight (corpus : doc = 1:1) | 0.310 (−10.1%) | 0.457 (−9.6%) | 0.342 (−9.7%) | 0.576 (−8.5%) |

## Component ablation

Having established that RAGSpot outperforms the baselines, we now ask *which* of its components accounts for the gain. RAGSpot is built from two core design elements: corpus filtering ("Corpus filtering"), which prunes the corpora to be searched, and rank-based fusion ("Rank-based fusion"), which merges the per-corpus results into a single ranking. We ablate in two steps — first which of the two stages drives accuracy, then how the fusion combines its two rank signals — on the single-query workload ($n$ = 2,237 queries) under the protocol of "Experiment setting", reporting point estimates and relative changes with respect to full RAGSpot over the same query set.

**Stage ablation.** We first remove each stage in turn (Table 6). *w/o corpus filtering* disables the gap-ratio selection so that every corpus is searched, which removes the corpus filtering stage together with its cluster-metadata scoring and radius normalization, while the rank-based fusion is left unchanged; *w/o rank-based fusion* keeps the same corpus selection but replaces the two-level rank fusion with a direct merge of the raw local similarity scores. The two stages play very different roles. Removing rank-based fusion is catastrophic, reducing nDCG@10 by 76.9% (from 0.345 to 0.079), which confirms that rank-based fusion is the primary source of RAGSpot 's accuracy. Removing corpus filtering, in contrast, leaves retrieval quality essentially unchanged (within 0.6% on all metrics) but forces every query to search all six corpora rather than 4.54 on average; corpus filtering is therefore an efficiency stage that prunes roughly a quarter of the per-query search cost without sacrificing accuracy. Radius normalization is a scale-calibration step inside this stage rather than an independent ranking signal, and its role is analyzed in "Corpus filtering".

**Rank-based fusion design.** Because rank-based fusion is what drives accuracy, we next isolate how its two rank signals — the corpus-level rank and the local document rank — should be combined (Table 7). *Corpus-rank-only* drops the local document rank; *local-rank-only* drops the corpus-level rank and is therefore equivalent to standard RRF over the document rankings; *equal weight* retains both signals but flattens their ratio to 1 : 1. The corpus-level rank is the dominant signal: local-rank-only (standard RRF) loses 42.6% nDCG@10, so the document ranks alone are far from sufficient. Dropping the local document rank instead (corpus-rank-only) costs only 1.8% , while flattening the weighting to 1 : 1 costs 10.1%, which supports the 1 : 5 ratio adopted in "Hyperparameter study" on empirical grounds rather than by convention.

## Query overhead

Each query to RAGSpot incurs two kinds of overhead — computational latency and communication volume — which we analyze in turn.

**Latency.** Figure 5 reports per-query end-to-end latency on both workloads. C-FedRAG achieves the lowest latency as it adopts a broadcast-based strategy without any routing step, incurring no additional computation for corpus selection. However, this comes at the cost of substantially lower retrieval quality, as all corpora are queried regardless of their relevance ("Retrieval accuracy"). RAGSpot introduces only a small overhead over

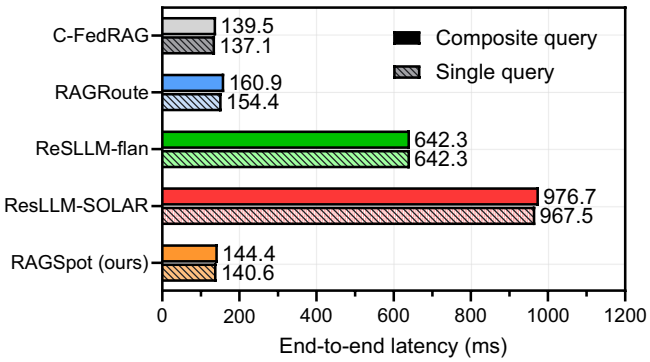



**Fig. 5** End-to-end query latency (ms) on the composite and single query workloads.

**Table 8** Communication overhead for query and retraining.

| Method | Query overhead (per query) | | | Retraining overhead(per corpus) |
|---|---|---|---|---|
| | Corpora | Comm. | nDCG@10 | Comm. |
| RAGSpot | 4.60 | 585 KB | 0.349 | 1.2 MB |
| RAGRoute | 1.90 | 241 KB | 0.284 | 2, 709.7 MB |
| C-FedRAG | 6.00 | 763 KB | 0.076 | — |
| ReSLLM-flan | 2.98 | 379 KB | 0.081 | — |
| ReSLLM-SOLAR | 3.30 | 419 KB | 0.161 | — |

C-FedRAG, with 3.51% higher latency on composite queries and 2.55% higher latency on single queries, while achieving substantially higher retrieval quality. This shows that, in our setting, lightweight routing adds only a small latency overhead relative to the broadcast baseline while improving corpus selection.

Compared to RAGRoute, RAGSpot is faster, reducing latency by 1.11× on composite queries and 1.10× on single queries. LLM-based routers are significantly slower: ReSLLM-flan is 4.45× slower on composite queries and 5.61× slower on single queries, while ReSLLM-SOLAR is 6.77× slower on composite queries and 6.88× slower on single queries. These results suggest that, despite their flexibility, LLM-based routing is impractical for latency-sensitive online serving. In summary, RAGSpot 's query latency is comparable to the broadcast baseline (within 3.5%), 1.10–1.11× faster than the training-based router, and 4.5–6.9× faster than LLM-based routers.

**Communication.** RAGSpot 's communication has two components, and we report both. *Query overhead (per query):* the routing query vector is sent to each selected corpus, and each returns its top-$k$ results, costing $b_q + k\,b_r = 127{,}096$ B per selected corpus ($b_q = 4{,}096$ B for a 1, 024-dimensional `float32` vector, $b_r = 4{,}100$ B per result, $k = 30$), independent of how many documents each corpus holds. Per-query communication therefore scales with the number of corpora a method selects (Table 8). RAGSpot selects 4.60 of the six corpora on average, giving $\approx$ 585 KB per query, about 77% of the broadcast upper bound ($N \times 127{,}096$ B). The learned and LLM-based routers send fewer bytes only by under-selecting, which lowers retrieval quality: the nDCG@10 column shows the cheaper routers reach at most 0.284 and broadcasting only 0.076, while RAGSpot attains 0.349 (full metrics in Tables 3–4).

*Retraining overhead (per corpus):* whenever a corpus joins, leaves, or updates its data, a training-based router must rebuild its training set by replaying the retrieval pipeline over the entire federation ("Retraining overhead"), using the same per-request byte model; this transfers 2, 709.7 MB at $N = 6$ (Figure 2b), and even unchanged corpora are re-queried. RAGSpot is training-free and avoids this entirely: only the affected corpus republishes its compact centroid–radius metadata (about 1.2 MB for our largest corpus), while the other corpora are untouched. A single reconfiguration event thus costs a training-based router as much communication as roughly 4, 600 of RAGSpot 's queries. The LLM-based routers and broadcasting incur no retraining communication either, since they require no training, and are therefore marked with a dash in Table 8; they nonetheless reach far lower retrieval quality and, for ReSLLM, much higher latency ("Query overhead"). Consequently, although a learned

**Table 9** One-time routing metadata generation overhead per corpus. The routing encoder $\omega$ (used to build the metadata and to encode routing queries) is shared across all corpora; only the local retrieval encoder $\phi_i$ differs.

| Corpus | Routing enc. $\omega$ | Local retrieval enc. $\phi_i$ | #Docs | $K$ | Time (s) |
|---|---|---|---|---|---|
| NFCorpus | UAE-Large-V1 | UAE-Large-V1 | 3,633 | 50 | 1.1 |
| SciFact | UAE-Large-V1 | gte-base | 5,183 | 71 | 1.3 |
| ArguAna | UAE-Large-V1 | UAE-Large-V1 | 8,674 | 93 | 3.1 |
| SciDocs | UAE-Large-V1 | mpnet | 25,657 | 160 | 5.3 |
| FiQA | UAE-Large-V1 | mpnet | 57,638 | 240 | 7.8 |
| Webis-Touche-2020 | UAE-Large-V1 | e5-base | 382,545 | 300 | 14.6 |

**Table 10** Sensitivity of $\gamma$ (restricted to $\gamma \geq 1$, which is required by construction; "Properties and complexity"). The selected hyperparameter value is highlighted in gray.

| $\gamma$ | nDCG@10 | Recall@10 | # Selected corpora | Routing F1 |
|---|---|---|---|---|
| 1.0 | 0.21 | 0.32 | 1.00 | 0.58 |
| 1.1 | 0.30 | 0.46 | 2.23 | 0.75 |
| 1.3 | 0.35 | 0.51 | 4.60 | 0.63 |
| 1.5 | 0.35 | 0.51 | 5.68 | 0.51 |

router such as RAGRoute transmits fewer bytes per query, that advantage is overturned by the recurring reconfiguration cost that RAGSpot eliminates by design.

## Metadata generation overhead

RAGSpot requires each corpus to generate centroid–radius metadata once, prior to serving queries. This step corresponds to the metadata generation process described in "Metadata generation", where each corpus performs document embedding and $k$-means clustering to summarize its semantic distribution. The routing metadata of every corpus is generated with the shared routing encoder $\omega$; importantly, each corpus remains free to use its own *local retrieval* encoder $\phi_i$, reflecting the heterogeneous nature of real-world FedRAG deployments.

Table 9 reports the wall-clock time for this one-time preprocessing step across all corpora, along with the local embedding model and the number of clusters used. We assume that each embedding model is preloaded for its corresponding corpus; thus, loading delay is not considered. Overall, metadata construction is efficient. Even for the largest corpus (Webis-Touche-2020 with 382,545 documents and 300 clusters), the total preprocessing time is only 14.6 s.

In our setting, this overhead is small for two reasons. First, metadata construction is a one-time cost amortized over all subsequent queries. Second, the resulting metadata is highly compact ($O(K \cdot e)$ per corpus), requiring orders of magnitude less storage and communication than document-level representations. These results demonstrate that RAGSpot 's metadata generation introduces minimal overhead while enabling efficient and accurate routing.

## Hyperparameter study

We analyze the sensitivity of RAGSpot to three key hyperparameters: the gap ratio $\gamma$, the RRF constant $\eta$, and the fusion weights, while keeping all other parameters fixed to their default values.

**Gap ratio $\gamma$ .** The gap ratio controls how aggressively RAGSpot prunes corpora whose match score is far behind the top-ranked corpus ("Corpus selection"). We evaluate its impact using nDCG, Recall, the number of selected corpora per query, and Routing F1, which is defined as the F1 score between the set of selected corpora and the ground-truth relevant corpora for each query. By construction, $\gamma \geq 1$ guarantees that the minimum-score corpus is always selected ("Properties and complexity"); we therefore restrict the sweep to $\gamma \geq 1$, since $\gamma < 1$

**Table 11** Sensitivity to RRF constant $\eta$. The selected hyperparameter value is highlighted in gray.

| $\eta$ | nDCG@10 | nDCG@30 | Recall@10 | Recall@30 |
|---|---|---|---|---|
| 10 | 0.344 | 0.385 | 0.486 | 0.631 |
| 20 | 0.353 | 0.384 | 0.520 | 0.633 |
| 40 | 0.348 | 0.382 | 0.508 | 0.631 |
| 60 | 0.348 | 0.381 | 0.508 | 0.627 |
| 80 | 0.348 | 0.380 | 0.508 | 0.625 |
| 100 | 0.348 | 0.380 | 0.508 | 0.625 |

**Table 12** Sensitivity to fusion weights ($w_{\text{local}} : w_{\text{global}}$). The selected hyperparameter value is highlighted in gray.

| $\mathbf{w}_{\text{local}} : \mathbf{w}_{\text{global}}$ | nDCG@10 | nDCG@30 | Recall@10 | Recall@30 |
|---|---|---|---|---|
| 5:1 | 0.239 | 0.286 | 0.384 | 0.551 |
| 2:1 | 0.260 | 0.299 | 0.414 | 0.555 |
| 1:1 | 0.310 | 0.343 | 0.452 | 0.573 |
| 1:2 | 0.335 | 0.364 | 0.488 | 0.597 |
| 1:5 | 0.348 | 0.381 | 0.508 | 0.627 |
| 1:10 | 0.346 | 0.384 | 0.489 | 0.633 |

would reject every corpus and yield an empty selection by definition rather than as an empirical finding. For $\gamma \geq 1$ , increasing $\gamma$ admits more corpora (the number of selected corpora grows from 1 to 5.68). Routing F1 peaks at $\gamma = 1.1$, but retrieval quality (nDCG and Recall) continues to improve until $\gamma = 1.3$ (by 16.7% and 10.9%, respectively). Beyond $\gamma = 1.3$, gains in nDCG and Recall saturate while the number of selected corpora continues to increase, leading to unnecessary routing overhead. We therefore adopt $\gamma = 1.3$ as the default, as it achieves a favorable balance between retrieval effectiveness and routing efficiency.

**RRF constant $\eta$ .** Table 11 shows that retrieval quality is largely insensitive to the choice of $\eta$ over a wide range. Although smaller values (e.g., $\eta = 20$) yield slightly higher nDCG@10 and Recall@10, the gains are marginal (within 0.01–0.02), suggesting that the effect of $\eta$ is limited in practice. As $\eta$ increases, performance quickly saturates, with nearly identical results for $\eta \geq 40$. This behavior is consistent with the role of $\eta$ as a smoothing parameter: smaller values emphasize top-ranked documents more aggressively, whereas larger values reduce rank sensitivity and promote more balanced aggregation. Given the negligible performance differences and the increased robustness at larger $\eta$, we adopt $\eta = 60$ as a stable default setting, which is also commonly used in prior RRF-based methods[28,29].

**Fusion weights.** Table 12 examines the relative weighting of the local (document-level) and global (corpus-level) ranks through the fusion weights $w_{\text{local}} : w_{\text{global}}$ (see "Hyperparameter study"). Configurations that overweight the local rank (e.g., 5 : 1) substantially underperform, indicating that the global corpus-level rank is essential for coordinating evidence across heterogeneous corpora. As the weight of the global rank increases, retrieval quality improves consistently, highlighting the importance of cross-corpus relevance. While increasing the global weight continues to improve deeper ranking metrics (nDCG@30 and Recall@30), it provides limited or diminishing gains for early precision (nDCG@10 and Recall@10). We therefore adopt 1 : 5 as the default, as it offers a strong balance between top-ranked accuracy and overall retrieval performance.

## Larger federation

The evaluation above uses six corpora. To assess whether RAGSpot remains effective as the federation widens, we repeat the evaluation on a federation of twelve BEIR corpora, twice the original number. The corpora are NFCorpus, SciFact, ArguAna, SciDocs, FiQA, Webis-Touche-2020, NQ, HotpotQA, DBpedia-Entity, FEVER, Climate-FEVER, and MS MARCO. To keep indexing tractable across twelve corpora, each corpus is sampled uniformly at random to at most 50,000 documents. All other conditions follow "Experiment setting": every method is evaluated on the same test queries, with the same local retrieval depth, the same metrics, and the same hardware. The single-corpus workload contains 13,937 queries and the composite workload 3,213 queries.

Because a retrieval-augmented generator consumes several documents as context rather than a single one, we report cutoffs $k \in \{5, 10, 30\}$ for this experiment.

**Results.** Table 13 shows that RAGSpot 's advantage persists as the federation widens. On single-corpus queries it reaches nDCG@10 = 0.475, against 0.411 for RAGRoute (1.16×), 0.191 for ReSLLM-SOLAR (2.49×), and 0.190 for C-FedRAG (2.50×); the corresponding Recall@10 margins are larger still (1.66×, 3.20×, and 3.15×). On composite queries, where a correct router must select multiple relevant corpora, RAGSpot improves nDCG@10 by 1.31× over RAGRoute and 1.19× over C-FedRAG. RAGSpot also remains the fastest routing method at 47 ms per query; broadcasting is comparable (48 ms) only because it performs no routing computation at all, at the cost of querying every corpus. Widening the federation therefore does not erode RAGSpot 's effectiveness or its cost advantage, although we limit this claim to the evaluated range rather than asserting that it extends to federations of arbitrary size.

## End-to-end generation quality

The evaluation so far measures routing and retrieval quality. We now examine whether RAGSpot 's retrieval advantage carries through to the generated answer. On the twelve-corpus federation ("Larger federation"), each routing method's top-10 retrieved documents are assembled into a common prompt, with context blocks numbered [1] … [$k$], and passed to a single fixed generator, GPT-4o-mini, which we run at temperature 0 and cap at 300 output tokens, so that only the routing method differs. We sample up to 100 eligible single-corpus queries from each of the twelve corpora using a fixed random seed (42). Because some corpora contain fewer than 100 eligible queries after filtering, this results in 824 single-corpus queries in total. All routers are evaluated on the same set of 824 queries. We also evaluate on all 3, 213 composite queries. The sampling script and the resulting query-id list are included in the deposited code (see Code availability). Because BEIR provides relevance judgments but no reference answers, reference-based answer accuracy (e.g., exact match) is undefined on these corpora; we therefore report two reference-free RAGAS metrics, both judged by GPT-4o-mini: *answer relevancy*, the cosine similarity (in the `text-embedding-3-small` embedding space) between the query and questions regenerated from the answer, and *context utilization*, which measures how effectively the retrieved context is used in the answer.

Table 14 shows that the routing differences propagate to generation. RAGSpot attains the highest answer relevancy and context utilization on both workloads. On single-corpus queries its answer relevancy is 0.502, against 0.381 for the best baseline (RAGRoute), a 1.32× margin; on composite queries it is 0.361, against 0.263 for the best baseline (C-FedRAG), a 1.37× margin. Context utilization follows the same ordering. Better corpus selection therefore yields not only higher retrieval scores but also more relevant and better-grounded generated answers.

**Table 13** Retrieval accuracy and query latency on the larger federation (twelve-corpus federation).

| Method | nDCG@$k$ | | | Recall@$k$ | | | Latency |
|---|---|---|---|---|---|---|---|
| | k = 5 | k = 10 | k = 30 | k = 5 | k = 10 | k = 30 | (ms) |
| *Single-corpus queries* | | | | | | | |
| RAGSpot | 0.419 | 0.475 | 0.505 | 0.551 | 0.708 | 0.812 | 47 |
| RAGRoute | 0.404 | 0.411 | 0.418 | 0.408 | 0.426 | 0.450 | 110 |
| C-FedRAG | 0.184 | 0.190 | 0.198 | 0.205 | 0.225 | 0.257 | 48 |
| ReSLLM-flan | 0.173 | 0.179 | 0.186 | 0.192 | 0.210 | 0.237 | 302 |
| ReSLLM-SOLAR | 0.185 | 0.191 | 0.198 | 0.202 | 0.221 | 0.247 | 792 |
| *Composite queries* | | | | | | | |
| RAGSpot | 0.158 | 0.170 | 0.192 | 0.113 | 0.165 | 0.247 | 51 |
| RAGRoute | 0.131 | 0.130 | 0.137 | 0.073 | 0.090 | 0.124 | 117 |
| C-FedRAG | 0.141 | 0.143 | 0.153 | 0.069 | 0.086 | 0.122 | 49 |
| ReSLLM-flan | 0.109 | 0.111 | 0.117 | 0.050 | 0.063 | 0.089 | 297 |
| ReSLLM-SOLAR | 0.113 | 0.114 | 0.122 | 0.049 | 0.063 | 0.094 | 805 |

**Table 14** End-to-end generation quality.

| Method | Single-corpus | | Composite | |
|---|---|---|---|---|
| | Answer relevancy | Context utilization | Answer relevancy | Context utilization |
| RAGSpot | 0.502 | 0.415 | 0.361 | 0.248 |
| RAGRoute | 0.381 | 0.394 | 0.215 | 0.224 |
| C-FedRAG | 0.323 | 0.334 | 0.263 | 0.231 |
| ReSLLM-flan | 0.286 | 0.271 | 0.188 | 0.177 |
| ReSLLM-SOLAR | 0.306 | 0.313 | 0.199 | 0.181 |

**Table 15** Concatenated versus naturally phrased composite workloads.

| Method | nDCG@$k$ | | | Recall@$k$ | | |
|---|---|---|---|---|---|---|
| | k = 5 | k = 10 | k = 30 | k = 5 | k = 10 | k = 30 |
| *Concatenated (synthetic stress test)* | | | | | | |
| RAGSpot | 0.151 | 0.157 | 0.173 | 0.111 | 0.159 | 0.221 |
| RAGRoute | 0.124 | 0.119 | 0.122 | 0.088 | 0.110 | 0.138 |
| C-FedRAG | 0.067 | 0.062 | 0.063 | 0.041 | 0.047 | 0.056 |
| ReSLLM-flan | 0.056 | 0.053 | 0.054 | 0.034 | 0.043 | 0.055 |
| ReSLLM-SOLAR | 0.097 | 0.091 | 0.092 | 0.057 | 0.068 | 0.085 |
| *Naturally phrased (LLM-rewritten)* | | | | | | |
| RAGSpot | 0.132 | 0.140 | 0.164 | 0.086 | 0.130 | 0.213 |
| RAGRoute | 0.081 | 0.078 | 0.083 | 0.049 | 0.063 | 0.087 |
| C-FedRAG | 0.087 | 0.083 | 0.085 | 0.063 | 0.070 | 0.081 |
| ReSLLM-flan | 0.075 | 0.072 | 0.072 | 0.047 | 0.056 | 0.070 |
| ReSLLM-SOLAR | 0.100 | 0.094 | 0.095 | 0.059 | 0.070 | 0.089 |

## Naturally phrased composite queries

The composite workload in "Experiment setting" is built by concatenation, which yields exact two-source ground truth but does not resemble how a user would phrase a multi-domain request. To check that our conclusions do not depend on this construction, we build a naturally phrased variant of the same workload. Each of the 3,464 concatenated pairs is rewritten by an instruction-tuned language model (Qwen2.5-7B-Instruct, greedy decoding) into a single fluent question, with the prompt instructing the model to preserve both information needs and to drop neither entity nor constraint. The rewritten queries inherit the judgments of the pair they were derived from, so the two workloads differ only in surface form.

Rewriting can silently drop or alter one of the two intents, which would invalidate the inherited judgments, so we audit the result rather than assume it is faithful. In an initial random sample of 60 rewrites, both intents are preserved in 91.7% of cases (Wilson 95% confidence interval [81.9%, 96.4%]). We subsequently expanded the audit to 100 items; among these, 90 preserved both intents, 9 altered one intent into a different question — typically a short, ambiguous query absorbed into the topic of its longer partner — and 1 dropped an intent entirely.

**Results.** Table 15 compares the two workloads. RAGSpot ranks first on both, with nDCG@10 of 0.157 on the concatenated workload and 0.140 on the natural one. What changes is the ordering of the baselines. The learned router RAGRoute falls from second to fourth, its nDCG@10 dropping from 0.119 to 0.078, so RAGSpot 's margin over it widens from 1.32× to 1.80×. The prompting-based routers move the opposite way: ReSLLM-SOLAR rises from third to second, improving from 0.091 to 0.094, and ReSLLM-flan also holds up, improving from 0.053 to 0.072. This behavior is consistent with LLM-based routing being less sensitive to surface phrasing than the concatenated form, while RAGSpot 's margin over C-FedRAG narrows from 2.52× to 1.68×. RAGSpot nonetheless ranks first under this substantially reshuffled baseline ordering, and its own score falls rather than rises under rephrasing, so the outcome is not an artifact of the concatenated construction. The remaining absolute changes reflect the roughly 10% of rewrites whose judgments no longer apply.

We retain the concatenated workload as the primary composite result because its judgments are exact, and report the naturally phrased workload as a robustness check. We do not claim that the rewritten queries are equivalent to queries authored by users; establishing a multi-domain benchmark with native relevance judgments remains future work.

## Related work

**FedRAG Systems.** Recent work has begun to explore RAG in federated settings, where queries must be answered using multiple distributed knowledge sources. C-FedRAG[9] and FRAG[30] propose system frameworks that enable secure or collaborative retrieval across multiple organizations. While these approaches demonstrate the feasibility of federated RAG, they often rely on broadcast-based or heuristic routing strategies, which can lead to redundant retrieval and increased system overhead. Moreover, they introduce additional challenges such as handling heterogeneous corpora and adapting to dynamic federation environments.

Complementary to system design, FeB4RAG[15] studies federated search in the context of RAG and introduces a benchmark for evaluating resource selection strategies across heterogeneous corpora. Its findings highlight the importance of selecting relevant sources rather than naively querying all available corpora, motivating the need for effective query routing mechanisms.

**Query Routing.** To address the inefficiency of naive broadcasting, several works propose routing mechanisms that select a subset of relevant corpora for each query. ReSLLM[11] leverages a large language model to predict corpus relevance from query semantics, enabling training-free routing but requiring an LLM invocation for each query. RAGRoute[10] introduces a lightweight neural router that estimates corpus relevance using query embeddings and corpus-level statistics. Although efficient at inference time, it relies on supervised training and assumes a retrieval encoder shared across corpora, so the router must be updated when the corpus set changes. C-FedRAG[9] avoids router training by broadcasting queries to all corpora, but directly merging local similarity scores can produce poorly calibrated global rankings when corpora use independently optimized retrieval encoders. In contrast, RAGSpot combines cluster-level metadata and rank-based fusion to jointly support training-free routing, heterogeneous local retrieval encoders, and a dynamically changing set of corpora without router retraining.

**Distributed Information Retrieval.** Selecting relevant collections from distributed corpora has been extensively studied in federated search[32,33]. Classical resource selection methods such as CORI[33] and ReDDE[34] estimate corpus relevance using term-based statistics derived from document content. RAGSpot addresses the same problem but differs in three key aspects: (1) it operates on dense embeddings rather than term frequencies, enabling semantic matching; (2) it uses radius-normalized centroid distances instead of unigram language models, providing distribution-aware scoring; and (3) it employs a fusion mechanism that jointly considers corpus-level and document-level signals, whereas classical methods typically treat selection and merging as separate stages.

## Discussion

**Comparison with classical resource selection.** Resource selection has a long history in distributed information retrieval, and it is fair to ask how RAGSpot compares with it empirically rather than only conceptually. We therefore ran CORI[33] as a routing baseline under the protocol of "Experiment setting": CORI replaces only the corpus-selection stage, while local retrieval and rank fusion remain identical, so the difference isolates the selection signal. CORI requires each corpus to publish term-level statistics (document frequencies and collection size) instead of centroids, which is compatible with our data-locality constraint. On the single-corpus workload it attains nDCG@10 = 0.241, against 0.345 for RAGSpot in the same pipeline; on the composite workload the values are 0.089 and 0.157. The gap is consistent with CORI's lexical basis: term statistics cannot capture semantic

relatedness between a query and a corpus whose vocabulary differs, which is precisely the regime dense routing targets. A second family of classical selectors remains outside our setting: methods that estimate collection relevance from a centralized sample index built by drawing documents out of each collection, such as ReDDE[34] and its descendants[32]. Because those samples are the documents themselves, the method requires exactly the data movement that federated RAG is meant to avoid, so we compare against selectors whose published statistics stay at the metadata level.

**Shared routing space.** RAGSpot requires every corpus to generate its routing metadata using the same routing encoder $\omega$ ("Properties and complexity"). This is an explicit design precondition. Corpus routing requires a mechanism for making relevance comparable across corpora. Embedding-based routers such as RAGRoute obtain this comparability by assuming a retrieval encoder shared across corpora. In contrast, RAGSpot shares only a lightweight routing encoder $\omega$, while each corpus retains its independently selected retrieval encoder $\phi_i$, local index, raw documents, and document-level embeddings.

Each corpus runs $\omega$ offline when joining the federation or refreshing its metadata, and publishes only $K$ centroids and $K$ radii. This separation preserves local retrieval autonomy. In our setting, replacing independently selected local encoders with a uniform retrieval encoder reduces nDCG@10 by up to 5.41% and Recall@10 by up to 6.55% ("Accuracy degradation from uniform embedding"). Because $\omega$ is used only for corpus selection rather than document ranking, our results indicate that a lightweight shared representation is sufficient for this coarser routing decision. Its offline overhead is small, with metadata generation taking 14.6 s for our largest corpus of 382,545 documents ("Metadata generation overhead").

The requirement nevertheless remains a limitation. A corpus that cannot or does not wish to use $\omega$ cannot participate in the current design, and supporting corpora that publish metadata in mutually incomparable routing spaces remains an open problem.

**Privacy considerations.** RAGSpot reduces the amount of information shared between corpora and the central router compared to approaches[10] that require raw documents or full embedding matrices. Each corpus exposes only $K$ centroids and $K$ scalar radii—a summary of $O(K \cdot e)$ floats rather than per-document embeddings. To clarify the information exposed during corpus routing, the central router receives only the published metadata, namely $K$ centroids and $K$ scalar radii per corpus. It does not receive the full document collection, per-document embeddings, or the local vector index for corpus selection. However, these aggregate statistics may still reveal coarse corpus characteristics; for example, an observer may infer that a corpus covers biomedical topics. We therefore make no claim of differential privacy or resistance to document reconstruction, membership inference, or other inference attacks. After routing, each selected corpus returns its retrieved top-$k$ documents to the aggregator as required by the FedRAG workflow. Accordingly, the benefit provided by RAGSpot is reduced information disclosure during routing and preservation of the local corpus and index, rather than end-to-end privacy preservation. Compared with approaches that expose raw corpora or per-document embedding matrices for routing, RAGSpot requires only compact aggregate metadata. In settings where such aggregate statistics are themselves sensitive, calibrated noise may be added to the centroids as an orthogonal extension[35].

**Practical implications.** The training-free design of RAGSpot enables practical deployment in real-world FedRAG systems. Organizations can join or leave a federation by simply registering or removing their cluster metadata, without triggering retraining that disrupts system availability. This seamless integration is particularly valuable in domains such as healthcare and finance, where new data sources frequently emerge and data sharing is constrained. Moreover, by allowing each corpus to retain its own retrieval encoder, RAGSpot preserves organizational autonomy while supporting heterogeneous federations. This design is compatible with modern LLM serving frameworks (e.g., SGLang[36]), enabling integration with existing inference pipelines.

**Future directions.** Several directions can further extend RAGSpot. First, adaptive clustering strategies that dynamically adjust the number of clusters per corpus could better capture complex or multi-modal data distributions. Second, the current rank-based fusion can be extended with query-aware weighting or adaptive retrieval budgets to improve aggregation efficiency and effectiveness across corpora. Finally, incorporating lightweight

feedback signals may enable online refinement of routing decisions without retraining, improving adaptability in dynamic federated environments.

## Conclusion

We present RAGSpot, a training-free query routing framework for federated retrieval-augmented generation. By operating solely on cluster-level metadata, RAGSpot eliminates the need for supervised training and for a shared retrieval encoder: each corpus keeps its own retrieval model, and only a lightweight routing encoder is shared, enabling seamless integration of new corpora while preserving corpus autonomy. To support heterogeneous retrieval encoders, RAGSpot combines radius-normalized corpus scoring with a rank-based fusion mechanism that jointly considers document-level and corpus-level relevance. Our evaluation demonstrates that RAGSpot improves retrieval accuracy over the strongest training-based router by 1.23–1.48× in nDCG and 1.28–1.66× in Recall, and over the strongest LLM-based router by 1.63–2.01× in nDCG and 1.93–2.34× in Recall, while achieving comparable or even lower query latency. These results demonstrate that RAGSpot provides an effective and efficient foundation for query routing in federated RAG systems.

**Acknowledgements** Corresponding authors are Gyeongsik Yang and Yeonho Yoo.

**Author contributions** H.E. contributed to conceptualization, methodology, software, validation, formal analysis, and writing – review and editing. Y.Y. contributed to conceptualization, methodology, writing – original draft, writing – review and editing, and supervision. G.Y. contributed to conceptualization, methodology, supervision, writing – review and editing, and project administration. All authors reviewed the manuscript.

**Funding** This research was partly supported by National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2024-00336564), by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by MSIT (RS-2024-00405128), by ICT Creative Consilience Program by IITP grant funded by MSIT (IITP-2026-RS-2020-II201819), and by IITP under the Artificial Intelligence Convergence Innovation Human Resources Development (IITP-2026-RS-2023-00254592) grant funded by the Korea government (MSIT).

**Data availability** All source corpora analyzed in this study are publicly available through the BEIR benchmark. The six-corpus federation uses SciFact, ArguAna, FiQA, NFCorpus, SciDocs, and Webis-Touche-2020; the twelve-corpus federation (§5.7) additionally uses NQ, HotpotQA, DBpedia-Entity, FEVER, Climate-FEVER, and MS MARCO. No new primary datasets were generated. Three query workloads are derived from these corpora: (i) the concatenated composite-query workload (§5.1); (ii) the naturally phrased composite-query workload (§5.9), obtained by rewriting (i) with Qwen2.5-7B-Instruct under greedy decoding; and (iii) the twelve-corpus workload (§5.7), whose documents are sampled to at most 50,000 per corpus and whose generation subset contains 824 single-corpus queries (§5.8). The scripts and random seeds that construct all three, together with the rewriting prompt, are deposited in the Zenodo record listed under Code availability, so that the workloads can be rebuilt from the BEIR corpora with the same procedure and parameters. The cluster metadata and the trained router checkpoints are deposited in the same record. The RAGSPOT implementation, the baseline routers (C-FedRAG, RAGRoute, and ReSLLM), the configuration files and random seeds of the experiments, and the workload-construction scripts (composite-query concatenation, twelve-corpus sampling, and LLM-based query rewriting with its prompt) are archived at Zenodo under https://doi.org/10.5281/zenodo.22822792.

**Code availability** RAGSpot implementation, the baseline routers (C-FedRAG, RAGRoute, and ReSLLM), the configuration files and random seeds of the experiments, and the workload-construction scripts (composite-query concatenation, twelve-corpus sampling, and LLM-based query rewriting with its prompt) are archived at Zenodo under https://doi.org/10.5281/zenodo.22822792.

## Declarations

**Competing interests** The authors declare no competing interests.

**Open Access** This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by-nc-nd/4.0/.

## References

1. Kamalloo, E., Dziri, N., Clarke, C. & Rafiei, D. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: long papers)*, 5591–5606 (2023).
2. Thoppilan, R. et al. Lamda: Language models for dialog applications. arXiv preprint arXiv:2201.08239 (2022).
3. Pan, L. et al. Fact-checking complex claims with program-guided reasoning. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*, 6981–7004 (2023).
4. Ji, Z. et al. Survey of hallucination in natural language generation. *ACM Comput. Surv.* **55**, 1–38 (2023).
5. Huang, L. et al. A survey on hallucination in large language models: Principles, taxonomy, challenges, and open questions. *ACM Transactions on Information Systems* **43**, 1–44 (2025).
6. Lewis, P. et al. Retrieval-augmented generation for knowledge-intensive NLP tasks. *In Advances in Neural Information Processing Systems* **33**, 9459–9474 (2020).
7. Guu, K., Lee, K., Tung, Z., Pasupat, P. & Chang, M. REALM: Retrieval-augmented language model pre-training. In *Proceedings of the 37th International Conference on Machine Learning*, 3929–3938 (PMLR, 2020).
8. Chakraborty, A., Dahal, C. & Gupta, V. Federated retrieval-augmented generation: A systematic mapping study. arXiv preprint arXiv:2505.18906 (2025).
9. Addison, P. C-fedrag: A confidential federated retrieval-augmented generation system (2024) arXiv:2412.13163 arXiv preprint.
10. Guerraoui, R. Efficient federated search for retrieval-augmented generation. In *Proceedings of the 5th Workshop on Machine Learning and Systems* 74–81 (2025).
11. Wang, S., Zhuang, S., Koopman, B. & Zuccon, G. Resllm: Large language models are strong resource selectors for federated search. *In Companion Proceedings of the ACM on Web Conference* **2025**, 1360–1364 (2025).
12. Lee, J. et al. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics* **36**, 1234–1240 (2020).
13. Chalkidis, I., Fergadiotis, M., Malakasiotis, P., Aletras, N. & Androutsopoulos, I. Legal-bert: The muppets straight out of law school. *In Findings of the association for computational linguistics: EMNLP* **2020**, 2898–2904 (2020).
14. Thakur, N., Reimers, N., Rücklé, A., Srivastava, A. & Gurevych, I. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks* (2021).
15. Wang, S., Khramtsova, E., Zhuang, S. & Zuccon, G. 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*, 763–773 (2024).
16. BAAI. Uae-large-v1: Universal text embeddings (2023).
17. Academy, A. D. Gte-base: General text embeddings (2023).
18. Reimers, N. & Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)* https://doi.org/10.18653/v1/d19-1410 (2019).
19. Wadden, D. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* 7534–7550 (2020).
20. Wachsmuth, H., Potthast, M., Köpsel, S. & Stein, B. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 40th European Conference on Information Retrieval (ECIR)* (2018).
21. Maia, M., Handschuh, S. & Freitas, A. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018 (WWW)* (2018).
22. Boteva, V. & Gurevych, I. A full-text learning to rank dataset for medical information retrieval. In *European Conference on Information Retrieval (ECIR)* (2016).

23. Cohan, A., Feldman, S., Beltagy, I., Downey, D. & Weld, D. S. Specter: Document-level representation learning using citation-informed transformers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)* (2020).
24. Bondarenko, A. Overview of touché 2020: Argument retrieval. In Working Notes of CLEF 2020 (2020).
25. Johnson, J., Douze, M. & Jégou, H. Billion-scale similarity search with gpus. *IEEE transactions on big data* **7**, 535–547 (2019).
26. Google. flan-t5-xl. https://huggingface.co/google/flan-t5-xl (2022). Accessed: 2026-04-12.
27. Changgil. K2s3-solar-11b-v2.0. https://huggingface.co/Changgil/K2S3-SOLAR-11b-v2.0 (2024). Accessed: 2026-04-12.
28. Cormack, G. V., Clarke, C. L. & Buettcher, S. 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 (2009).
29. Lin, J. Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations. In *Proceedings of the 44th international ACM SIGIR conference on research and development in information retrieval* 2356–2362 (2021).
30. Zhao, D. Frag: Toward federated vector database management for collaborative and secure retrieval-augmented generation (2024) arXiv preprint arXiv:2410.13272
31. Callan, J. Distributed information retrieval. In *Advances in Information Retrieval* 127–150 (Springer, 2002).
32. Shokouhi, M. & Si, L. Federated search. *Foundations and Trends in Information Retrieval* **5**, 1–102 (2011).
33. Callan, J., Lu, Z. & Croft, W. B. Distributed information retrieval ii: Collection retrieval. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (1995).
34. Si, L. & Callan, J. Relevant document distribution estimation method for resource selection. In *Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 298–305 (ACM, 2003).
35. Kulynych, B., Gomez, J. F., Kaissis, G., du Pin Calmon, F. & Troncoso, C. Attack-aware noise calibration for differential privacy. *Adv. Neural Inf. Process. Syst.* **37**, 134868–134901 (2024).
36. Zheng, L. et al. Sglang: Efficient execution of structured language model programs. *Advances in neural information processing systems* **37**, 62557–62583 (2024).

**Publisher's note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

## Authors and Affiliations

### Hosun Eoo[1] · Yeonho Yoo[2] · Gyeongsik Yang[3]

✉ Yeonho Yoo
yhyoo@dgu.ac.kr

✉ Gyeongsik Yang
g_yang@korea.ac.kr

[1] Department of Applied Artificial Intelligence, Korea University, Seoul 02841, Republic of Korea

[2] Department of Computer Science and Artificial Intelligence, Dongguk University, Seoul 04620, Republic of Korea

[3] Department of Computer Science and Engineering, Korea University, Seoul 02841, Republic of Korea