Efficient Federated Search for Retrieval-Augmented Generation

Rachid Guerraoui, Anne-Marie Kermarrec, Diana Petrescu,
Rafael Pires, Mathis Randl, Martijn de Vos

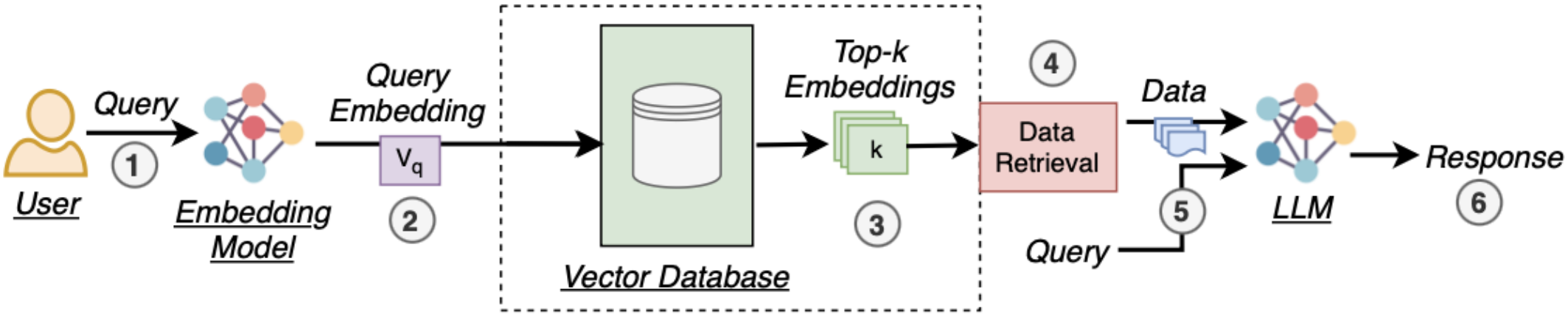
Why do we need RAG?

Reduces LLM hallucinations

Keeps responses up-to-date without retraining

Grounds LLM response in credible sources

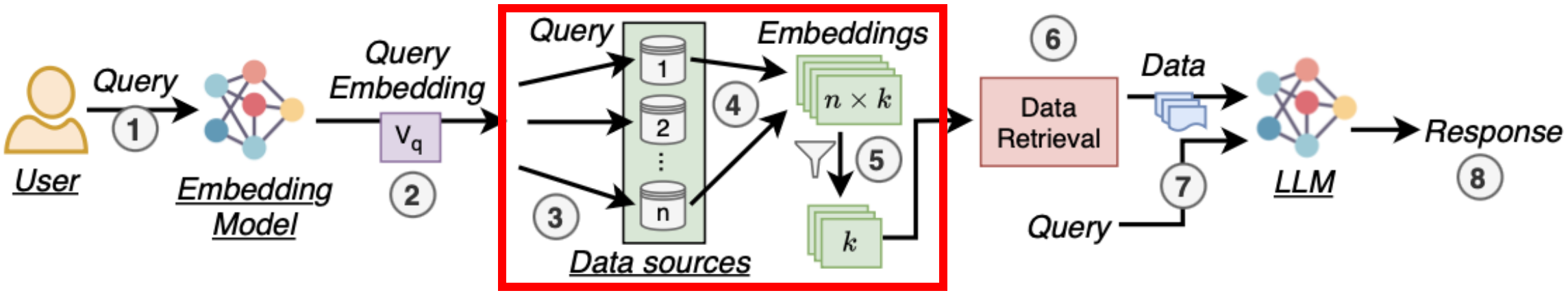
How does centralized RAG work?



A single vector database handles all queries - efficient, but **assumes all data is centrally stored**

Problem: Information is often **spread across multiple data sources**

Federated RAG

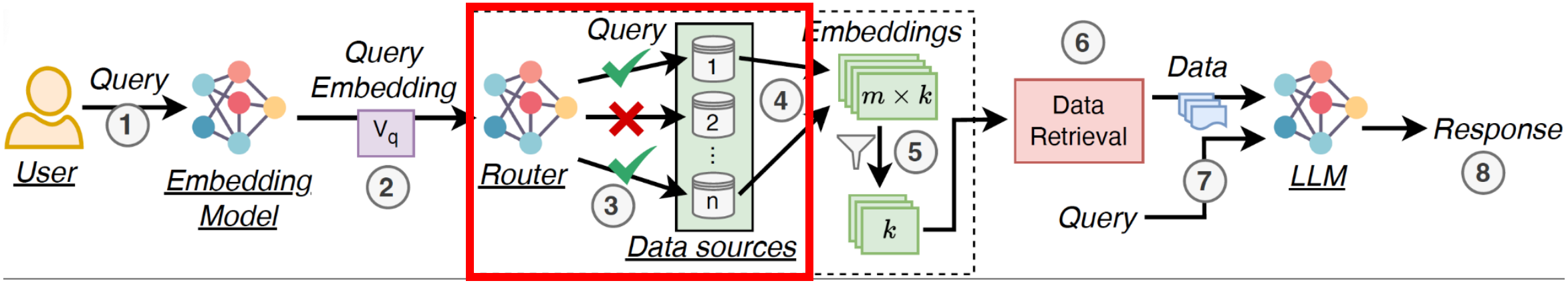


Enables **unified access** to multiple data sources in real time

Bypasses data migration - avoiding regulatory and technical barriers

Works with existing infrastructure - no need for major changes

Our contribution: RagRoute



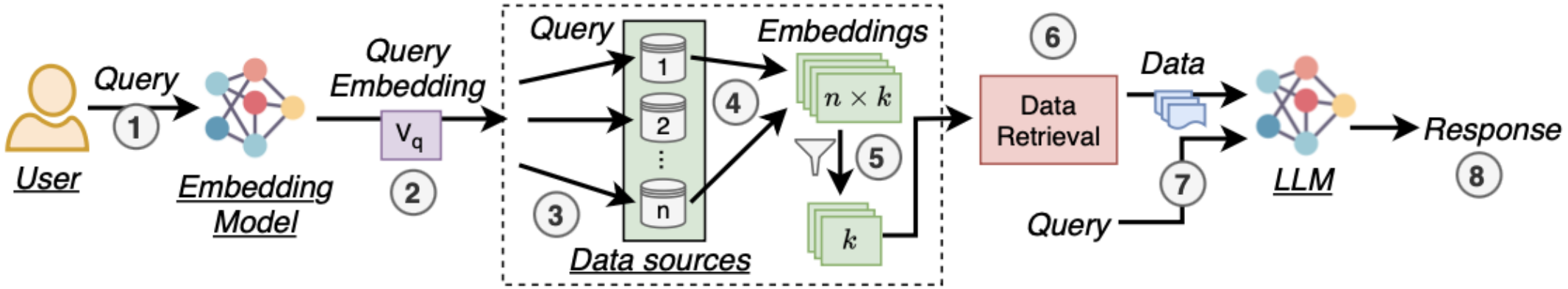
The **number of relevant sources depends on the query**

Querying **irrelevant sources** can **increase hallucinations**

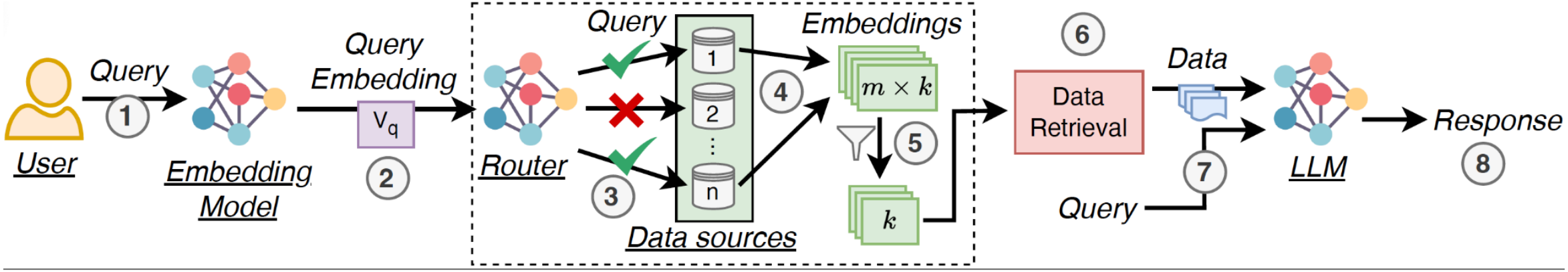
Unnecessary queries increase communication volume and compute cost

RagRoute training

Training



Inference



RagRoute architecture

Input Features

Query embedding
Dataset centroid
Query-centroid similarity
Number of documents
Dataset density

Model Architecture

Binary relevance indicator with 3-layer fully connected NN

Hidden Layer 1: 256 neurons
LayerNorm → ReLU → Dropout
Hidden Layer 2: 128 neurons
LayerNorm → ReLU → Dropout

Training Setup

Binary Cross-Entropy Loss

Positional weight for imbalance

Cyclic learning rate scheduler

30% train, 10% val, 60% test

Evaluation: MIRAGE benchmark

MIRAGE Benchmark		MEDRAG Corpora	
Dataset	**Size**	**Corpus**	**Chunks**
MMLU-Med	1,089	PubMed	23.9M
MedQA-US	1,273	StatPearls	301.2k
MedMCQA	4,183	Textbooks	125.8k
PubMedQA	500	Wikipedia	29.9M
BioASQ-Y/N	618	MedCorp (Fusion)	54.2M

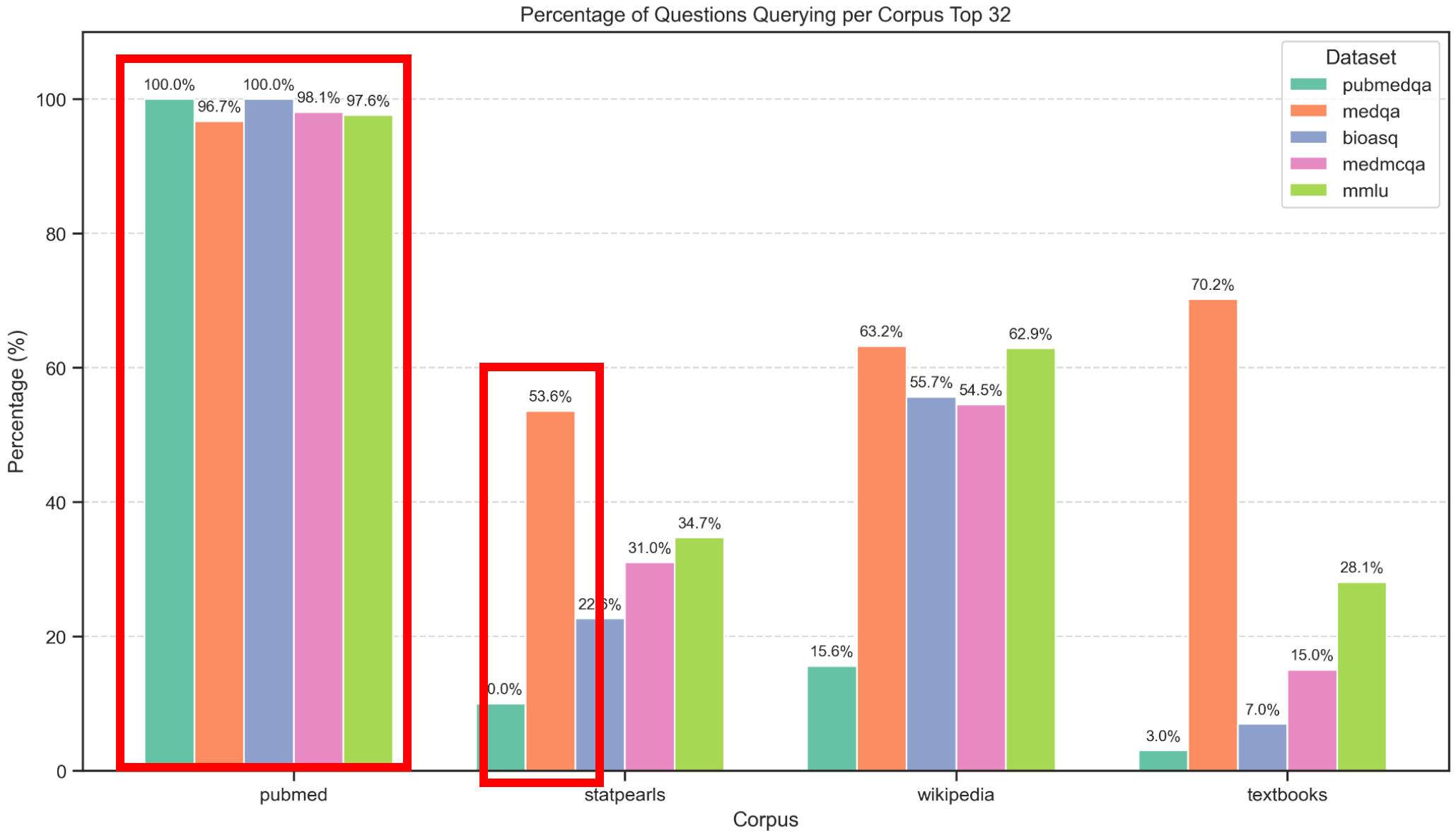
Data sources: we use each **MEDRAG corpus** as a data source in our setting

Evaluation: MMLU benchmark

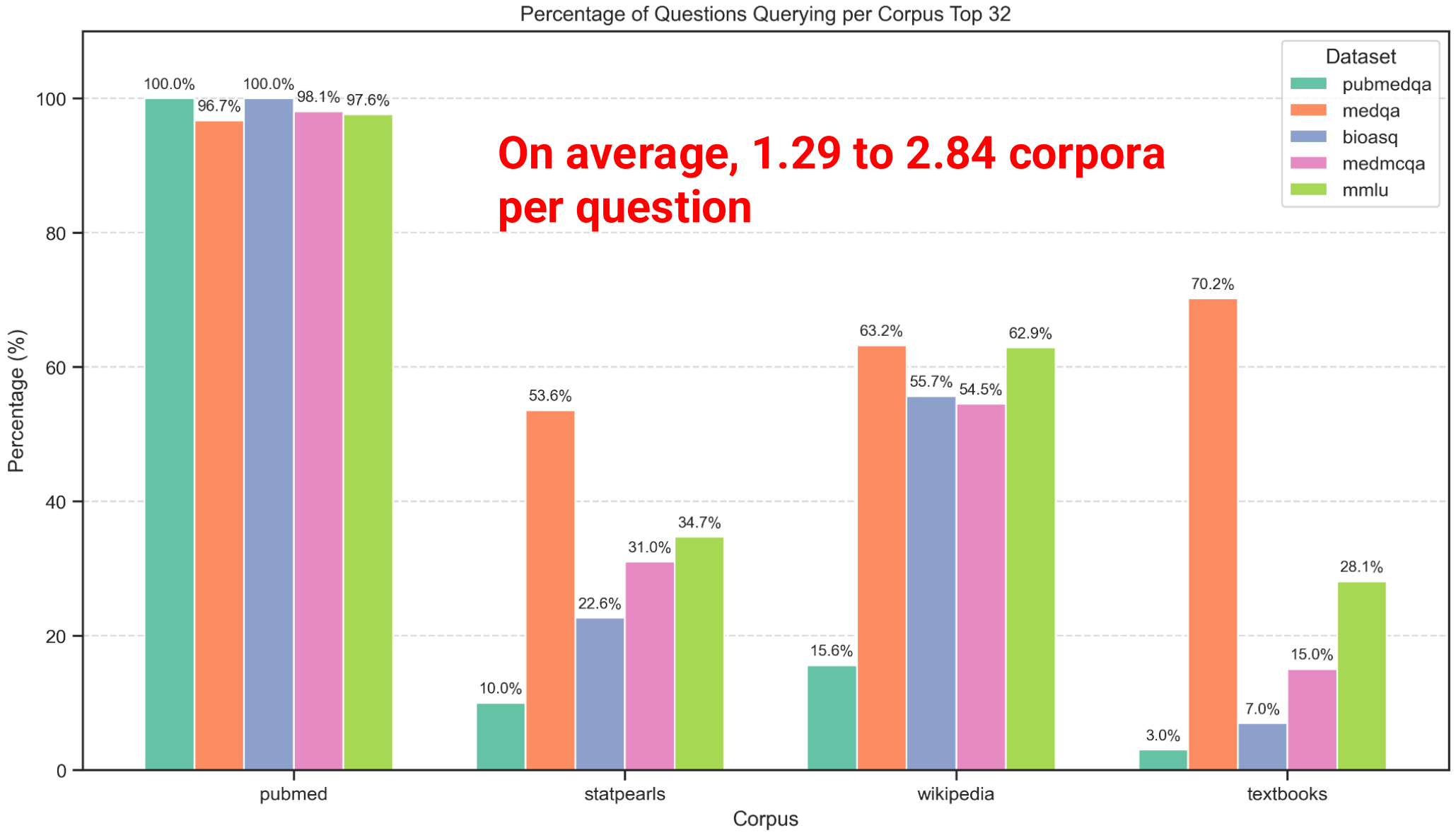
question string · *lengths*  41 243	subject string · *classes*  1 value	choices sequence · *lengths*  4 4	answer class label  4 classes
Find the degree for the given field extension Q(sqrt(2), sqrt(3), sqrt(18)) over Q.	abstract_algebra	["0", "4", "2", "6"]	1 B
Let p = (1, 2, 5, 4)(2, 3) in S_5 . Find the index of <p> in S_5.	abstract_algebra	["8", "2", "24", "120"]	2 C
Find all zeros in the indicated finite field of the given polynomial with coefficients in…	abstract_algebra	["0", "1", "0,1", "0,4"]	3 D
Statement 1 | A factor group of a non-Abelian group is non-Abelian. Statement 2 | If K is a…	abstract_algebra	["True, True", "False, False", "True, False", "False, True"]	1 B

Data sources: to simulate data sources, we group the embeddings of **Wikipedia snippets** into 10 clusters using the k-means algorithm.

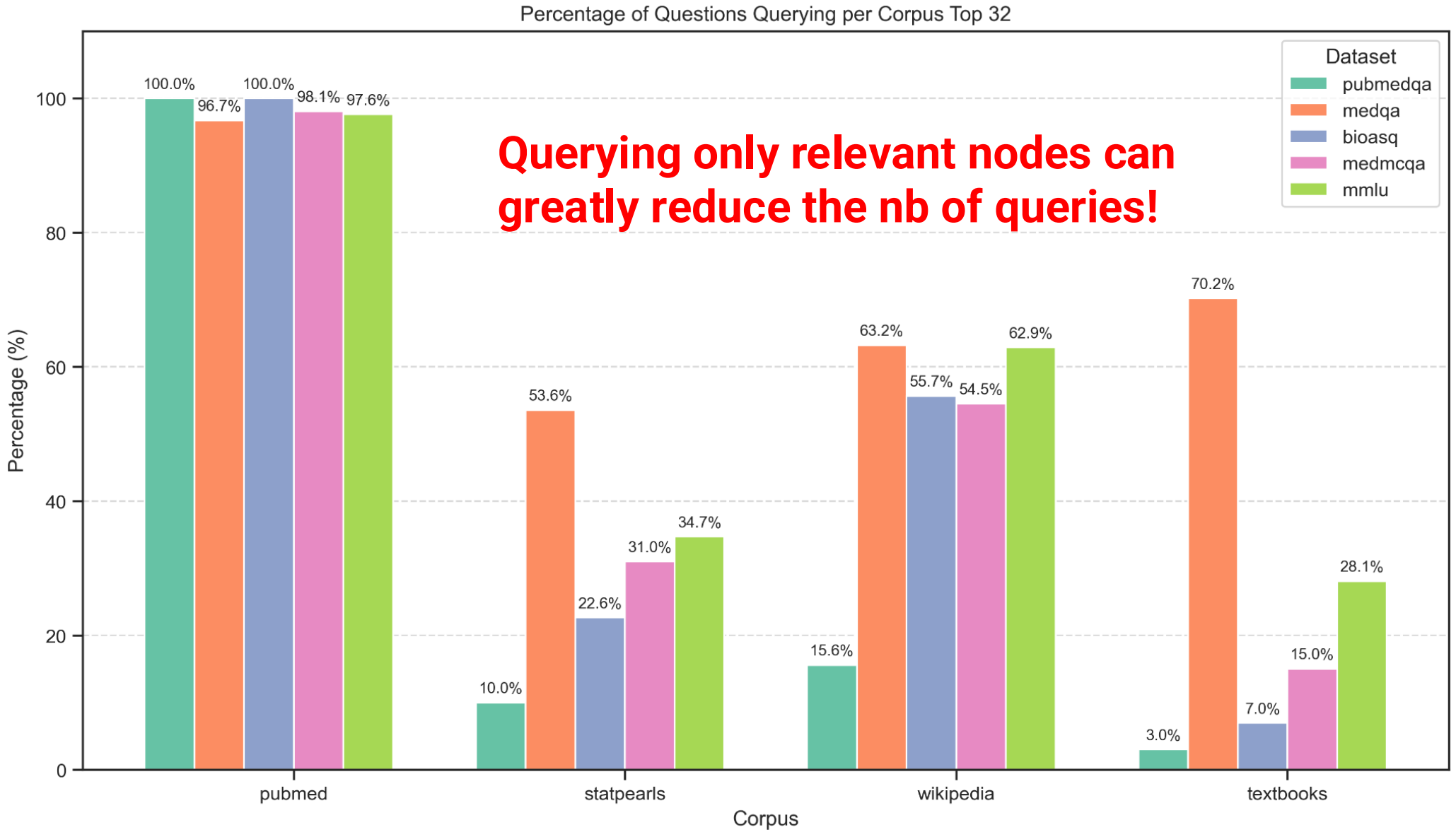
MedRAG document distribution top 32



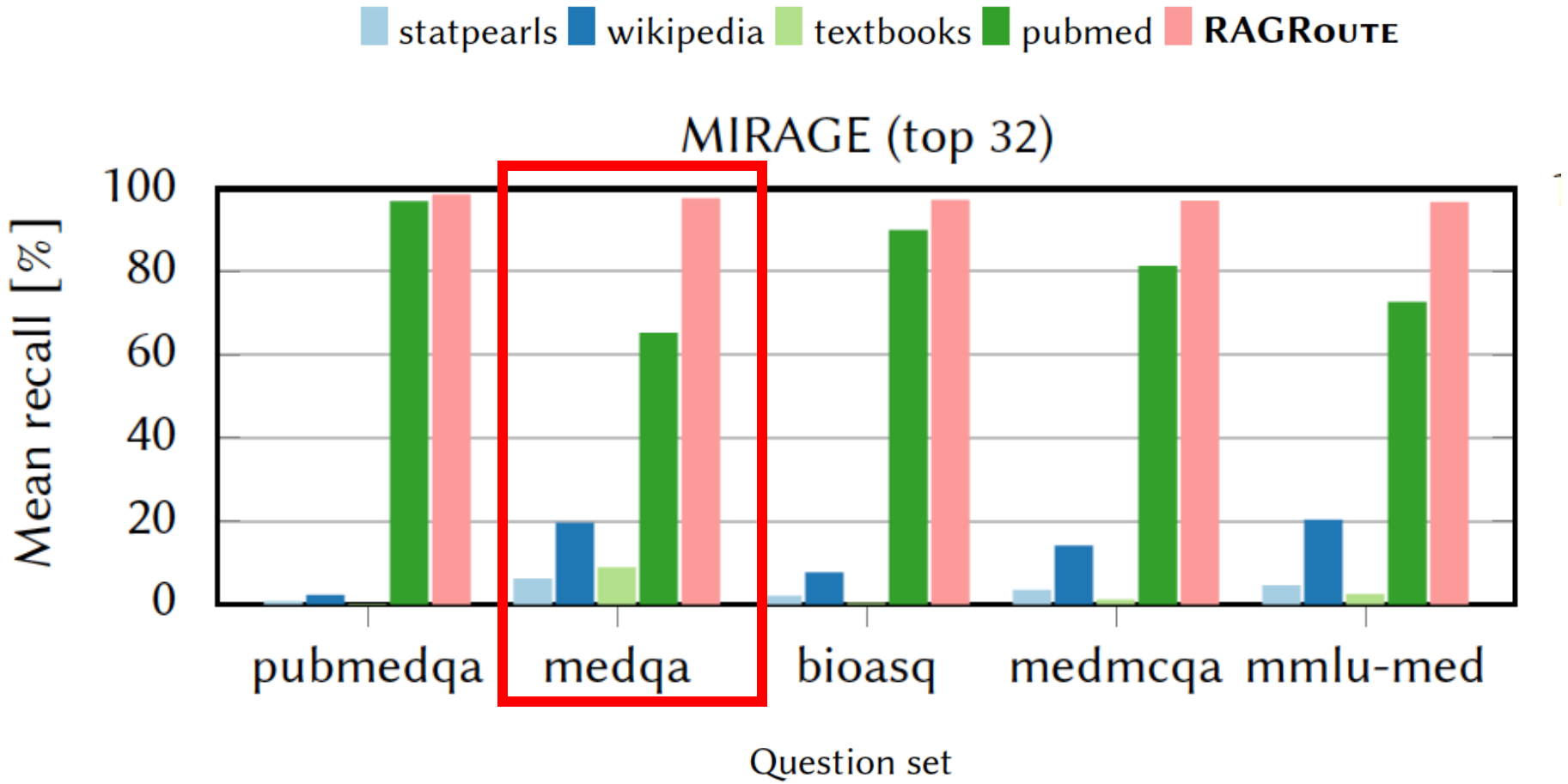
MedRAG document distribution top 32



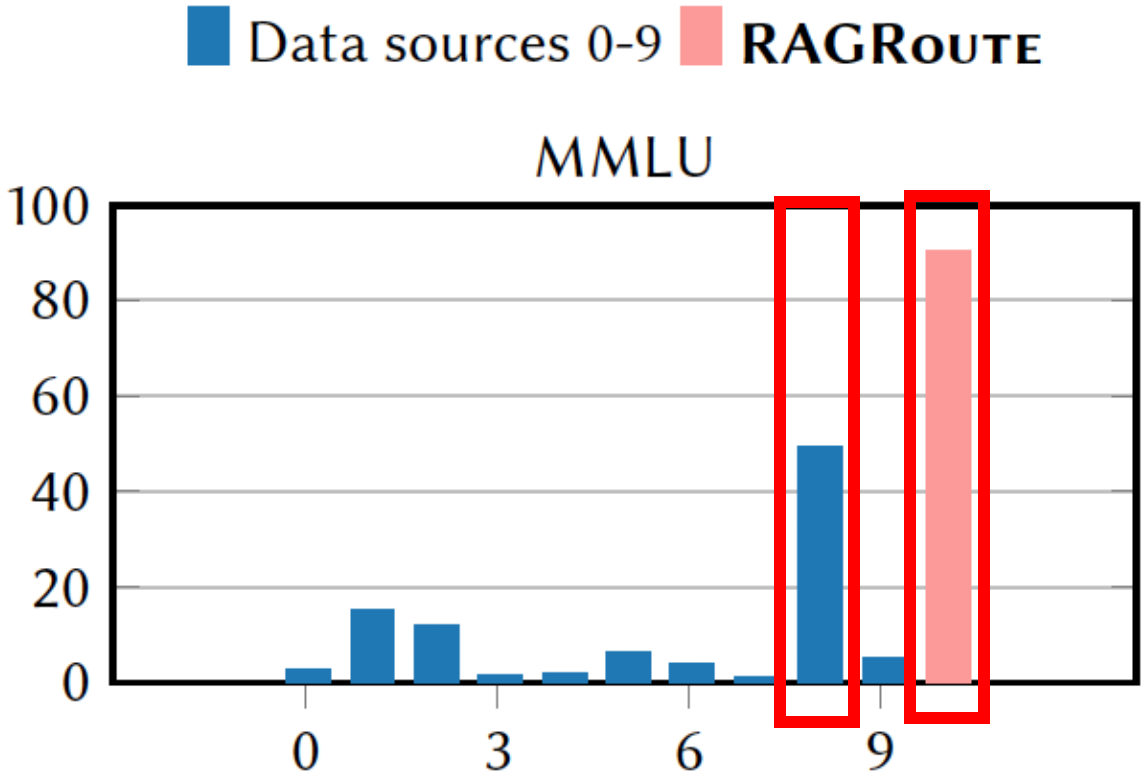
MedRAG document distribution top 32



Recall


statpearls
wikipedia
textbooks
pubmed
RAGRoute
MIRAGE (top 32)
Mean recall [%]
100
80
60
40
20
0
pubmedqa
medqa
bioasq
medmcqa
mmlu-med
Question set

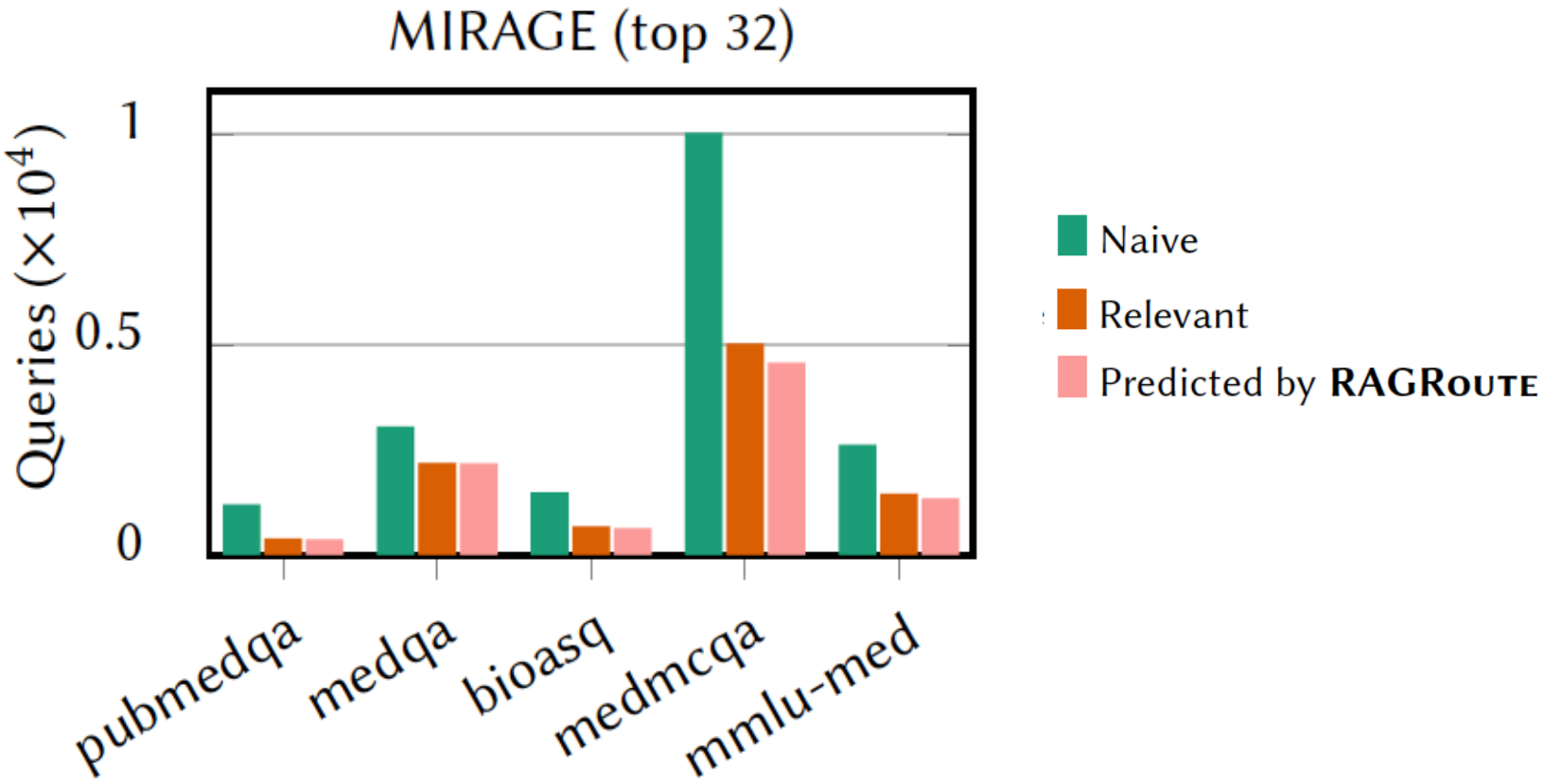
Recall



Classification results

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

Number of queries



Up to **71.3% reduction** for MIRAGE benchmark

77.5% reduction for MMLU benchmark (from 13 890 to to 3126)

End-to-end LLM accuracy
-
MIRAGE

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**

We use the **LLaMA 3.1 Instruct model**

Conclusion

RAGRoute

- Novel and efficient **routing mechanism** for federated RAG
- **Reduces** total number of **queries by up to 77.5%**
- **Maintains** high **retrieval quality** and end-to-end **accuracy**

Bonus

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

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