LEARNING WHAT RETRIEVAL MISSES: FEDERATED RETRIEVAL-RESIDUAL MEMORY FOR LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Retrieval-augmented language models rely on a retriever to supply what their parameters lack, but a retriever returns only its top-ranked passages, and part of the evidence in the very corpus it searches is never among them. Because a deployed retriever is fixed, the same evidence is missed every time and nothing in the system learns from the miss. Yet these persistent misses mark the knowledge a model should carry in its parameters. Memorizing documents indiscriminately spends capacity instead on knowledge that retrieval delivers reliably, and it can make the model answer from memory even where the retrieved context is already correct. We introduce Federated Retrieval-Residual Memory (FedR^2M), which uses the persistent misses of the fixed retriever as the criterion for what to memorize. Each data owner becomes a client and discovers its retrieval residuals, the questions whose supporting evidence its own corpus holds but its fixed retriever misses. The client then consolidates the missing knowledge into a low-rank adapter in three stages that write it, force the model to recall it from parameters, and protect the questions that retrieval already serves correctly. A server averages the client adapters into one global adapter, so memory built from one client's retrieval residuals serves all clients while raw data stays local, and inference pairs that global adapter with the unchanged retriever for every question, without per-question routing. Across four question-answering benchmarks, FedR^2M improves macro F1 by a relative 36% or more over every baseline in our comparison, gains 30% on retrieval hits and 59% on retrieval misses over retrieval-only prompting, and, without pooling raw data, matches centralized training of the same memory. The entire shared memory is one 13 MiB adapter, about a thousandth of the 13 GiB of per-passage adapters that document parameterization builds for the same questions.

1 INTRODUCTION

Retrieval-augmented generation connects a language model to an external corpus, so that knowledge the model lacks can be retrieved at question time rather than memorized in advance (Lewis et al., 2020; Karpukhin et al., 2020). A retriever returns only its top-ranked passages, so part of the evidence in the corpus never reaches the model. A deployed retriever is typically frozen, its ranking and index unchanged, so a question loses the same evidence every time it is asked, and nothing in the system learns from the miss. The knowledge is in the corpus, yet the model never uses it.

Two established responses address the problem from either end of the pipeline. At one end, improving the retriever targets the misses directly (Xu et al., 2026), but cost and stability keep a production retriever frozen, and any replacement is still a fixed retriever with misses of its own. At the other end, writing documents into model parameters (Su et al., 2025a; Zuo et al., 2026) proceeds indiscriminately, spending capacity on knowledge that retrieval delivers reliably and building memory that the model follows even where the retrieved context is already correct, and federated retrieval-augmented systems for corpora that cannot be pooled make the same choice when they share document adapters or distilled memories (Liang et al., 2026; Gao et al., 2026). Neither end reads the persistent misses of the deployed retriever as evidence of what the model itself lacks, so neither draws a boundary around the knowledge that needs to move into parameters.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

That boundary can be computed directly from the retriever's misses, with two checks that a data owner runs on each of its training questions, whether its corpus holds sufficient supporting evidence, and whether the fixed retriever exposes that evidence, or the answer itself, in the top-K. Questions that pass the first check and fail the second are the ones a model should memorize, and we call them retrieval residuals. Both checks depend only on the frozen retriever, the frozen corpus, and reference annotations, so the boundary is auditable. A grounding failure, where the evidence is retrieved but the model still answers incorrectly, is never counted as a retrieval residual.

Turning retrieval residuals into memory raises three difficulties. The set of retrieval residuals must be fixed before training, or the adapter chooses its own targets. New memory must not make the model stop answering from a context that is already correct, so it must be added without unlearning how to use reliable retrieval. Retrieval residuals are scattered across data owners whose corpora cannot be pooled, so memory built from one owner's misses must reach the others without moving passages, questions, or labels.

We propose Federated Retrieval-Residual Memory (FedR^{2}M), which turns the retrieval residuals of each data owner into one shared adapter. Each owner becomes a client. Retrieval residual discovery runs the two checks before any adapter exists and never consults the model's answers, so the training targets are fixed in advance. Parametric memory consolidation writes the missing evidence into a low-rank adapter, forces the model to recall it from parameters, and closes with a stage that protects retrieval hits. Federated adapter aggregation averages the client adapters on a server, so memory built at one client reaches every client while passages, questions, and labels never leave their owner, and global adapter inference serves every question with the unchanged retriever and the one global adapter, without per-question routing. On four open-domain question-answering benchmarks, FedR^{2}M improves macro F1 by at least 36% over every baseline in our comparison and by 48% over FedMosaic, which scores highest among the document-adapter systems. It improves on retrieval-only prompting by 30% on the retrieval hits and by 59% on the misses, whereas the document-adapter systems fall below it on the hits. It also matches centralized training of the same memory without pooling raw data and outperforms local training matched on total optimization. A blind test shows that the gain comes from both the boundary and the pairing of each question with its own missing evidence and answer. The one global adapter also outperforms per-passage document parameterization on the same questions and retrieved passages with about a thousandth of the storage.

Our contributions are the following.

- We formulate retrieval residuals, the questions whose evidence a local corpus holds but a fixed retriever misses, as an auditable boundary for what a retrieval-augmented model should memorize.
- We develop parametric memory consolidation, whose write, recall, and protect stages turn missed evidence into adapter memory while managing the trade-off with reliable retrieval through the stage order.
- We show that federated adapter aggregation outperforms local training matched on total optimization while raw data stays local, so knowledge learned from retrieval residuals transfers across clients.

2 Related Work

Retriever adaptation. Retrieval-augmented generation pairs a language model with a retriever over an external corpus (Lewis et al., 2020), typically a dense retriever (Karpukhin et al., 2020). One line of work trains the retriever with the language model (Guu et al., 2020; Izacard et al., 2023; Shi et al., 2024b), and another changes the process around a fixed retriever, rewriting the query (Ma et al., 2023), retrieving again under uncertainty (Jiang et al., 2023), interleaving retrieval with reasoning (Trivedi et al., 2023), learning when to retrieve and what to trust (Asai et al., 2024), or repairing the abstraction gap between question and evidence (Xu et al., 2026). Both lines deploy a fixed retriever with misses of its own, and neither records which questions it persistently misses.

Parametric knowledge injection. The complementary line writes knowledge into the model, by continued training, which stores a fact reliably only when it appears in many forms (Jang et al., 2022; Allen-Zhu & Li, 2024; Ovadia et al., 2024), by model editing (Meng et al., 2022), or by document

parameterization with low-rank adapters (Hu et al., 2022). Parametric RAG trains one adapter per document and loads the adapters of the retrieved documents at question time (Su et al., 2025a), DyPRAG generates the adapter from the document at test time (Tan et al., 2025), DistilledPRAG distills the retrieved passages into an adapter through a trained generator (Chen et al., 2026), and Poly-PRAG keeps a small routable set (Su et al., 2025b). Later work measures adapter capacity and composition, compares adapters with a compressed context, retrieves and merges adapters from a library, routes among parametric memories, or attributes output claims to the prompt or the parameters (Back et al., 2026; Zuo et al., 2026; Fleshman & Van Durme, 2025; Zhou & Zhang, 2026; Ji et al., 2026; Oestreich et al., 2026). In all of these, what enters the parameters is decided per document or per dataset, never by whether retrieval fails on a question.

Knowledge conflicts and continual adaptation. Writing knowledge into a model that also reads retrieved passages raises the question of which source prevails. Models answer from parametric memory against a contradicting passage (Longpre et al., 2021), accept coherent external evidence more readily (Xie et al., 2024), and gain from retrieval on long-tail facts while losing on facts they already know (Mallen et al., 2023). Context-aware decoding (Shi et al., 2024a) and GuarantRAG (Zhao et al., 2026) repair how the generator uses retrieved evidence, and continual learning protects a previous task by replay (Rolnick et al., 2019) or parameter penalties (Kirkpatrick et al., 2017) rather than the questions a retriever already answers correctly. None of these lines measures what new memory costs on retrieval hits.

Federated retrieval-augmented generation. Federated learning averages client updates on a server while the data stays local (McMahan et al., 2017), now usually as low-rank adapters for language models (Zhang et al., 2023; Sun et al., 2024). Retrieval-augmented generation over corpora with different owners either keeps every corpus as text and routes each query, to the silos a classifier selects (Shojaee et al., 2025), through a learned router over the silos (Guerraoui et al., 2025) or over silo-specific models (Zhang et al., 2025), or through an encrypted federated vector database (Zhao, 2024), or shares a trained artifact, as FedRAG (Fajardo et al., 2025), FedMosaic with its selectively aggregated document adapters (Liang et al., 2026), and FD-RAG with its distilled question-answer memory (Gao et al., 2026) do. In every case what a client retrieves or shares is determined by its documents, not by what its retriever fails to return, and privacy rests on data locality, which shared updates can undermine (Zhu et al., 2019) without secure aggregation (Bonawitz et al., 2017) or differential privacy (Abadi et al., 2016).

3 Federated Retrieval-Residual Memory

FedR^2M makes retrieval failures the criterion for what to memorize and does so in the four modules of Figure 1, with the complete procedure in Algorithm 1 of Appendix A. Retrieval residual discovery identifies, on each client, the questions whose supporting evidence the client's corpus holds but its fixed retriever misses. Parametric memory consolidation writes that missing knowledge into a low-rank adapter in three stages that protect the questions retrieval already serves. Federated adapter aggregation averages the client adapters on a server, so what one client learns reaches every client without moving raw data, and global adapter inference answers every question with the unchanged retriever and the single global adapter.

3.1 Retrieval residual discovery

Retrieval residual discovery labels, before any training, the questions whose supporting evidence is in the client's corpus but not in the fixed retriever's top-K. Client k holds a local corpus $\mathcal{D}_k$ and a set of local training questions $\mathcal{Q}_k$. A retriever R_K is fixed before training and is never updated. For a question q it returns an ordered context

$$\mathcal{C}_q = R_K(q, \mathcal{D}_k) \tag{1}$$

consisting of the K highest-scoring passages. Let a_q be the reference answer and let $\mathcal{E}_q$ be the annotated supporting sentences that jointly support a_q. We define two indicators,

$$h_q = \mathbf{1}[\mathcal{C}_q \text{ contains the answer or sufficient support for } a_q]\,, \tag{2}$$

$$v_q = \mathbf{1}[\mathcal{E}_q \text{ is available in } \mathcal{D}_k]\,, \tag{3}$$

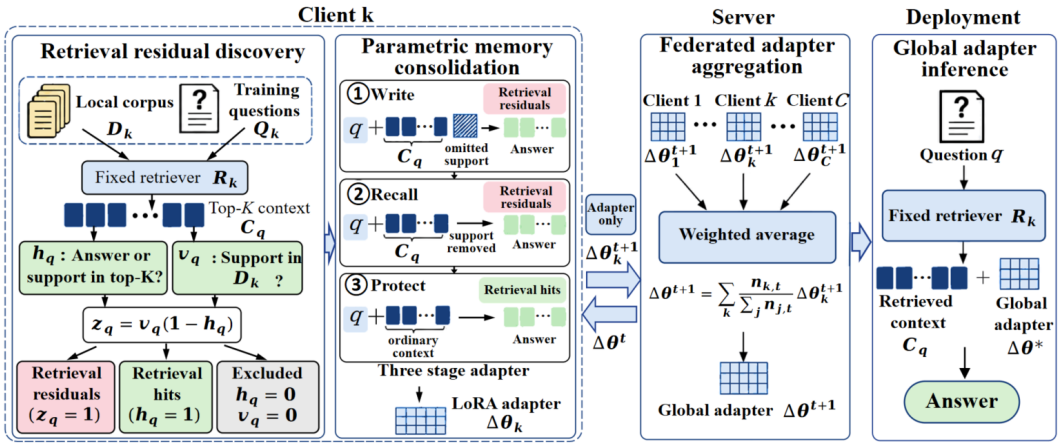

Figure 1: FedR^{2}M overview. Each client discovers its retrieval residuals, consolidates the missing knowledge into a LoRA adapter in the write, recall, and protection stages, and uploads only the adapter. The server averages client adapters each round; the final adapter serves every question alongside the unchanged retriever.

and call $h_q = 1$ a *retrieval hit*, $h_q = 0$ a *retrieval miss*, and v_q the *local availability* indicator. The *retrieval residual* indicator marks the questions whose support exists locally but is not retrieved,

$$z_q = v_q\,(1 - h_q). \tag{4}$$

For questions with $z_q = 1$, parametric memory is necessary, because retrieval will keep failing, and legitimate, because the client holds the knowledge. The label belongs to the question, not to a document, so FedR^{2}M selects what to memorize per question at each client.

The two indicators sort a client's questions into three groups, retrieval residuals with $z_q = 1$ that are memorized, retrieval hits with $h_q = 1$ that are replayed to protect how the model already uses reliable context, and questions with $h_q = 0$ and $v_q = 0$ that are excluded for lack of local evidence. A wrong answer by itself never marks a retrieval residual, because the boundary never consults the model's answers. Appendix A states the string-matching test that computes the labels.

3.2 Parametric Memory Consolidation

Parametric memory consolidation trains one LoRA adapter on three kinds of examples built from the discovered labels, in three stages, write, recall, and protect, that write the missing knowledge, force its recall, and protect the retrieval hits, so that the model keeps answering from a context that is already correct after the new memory is added. A retrieval residual yields a write example and a recall example, a retrieval hit yields a protection example, and every example asks for the short answer with only the answer tokens supervised.

Write, recall, and protection examples. The three example types share one template and differ in the evidence the input exposes and in the instruction line that opens it. With $\mathcal{E}_q \setminus \mathcal{C}_q$ denoting the supporting sentences whose passages retrieval omitted,

$$x_q^{\text{write}} = [\,q,\ \mathcal{C}_q,\ \mathcal{E}_q \setminus \mathcal{C}_q\,], \qquad x_q^{\text{recall}} = x_q^{\text{protect}} = [\,q,\ \mathcal{C}_q\,], \qquad y_q = a_q, \tag{5}$$

where write and recall examples come from retrieval residuals and protection examples from retrieval hits. A write example states in one sequence the question, the context the retriever provides, the supporting sentences it omits, and the answer. A recall example drops the omitted support, so the model can produce the answer only from its parameters. A protection example replays a retrieval hit with its ordinary context and keeps the model answering from that context after the new memory is added. Recall and protection examples are formally identical and differ only in whether $\mathcal{C}_q$ supports a_q. Inference presents the same two cases, and the same adapter must supply missing knowledge in the one and defer to the context in the other.

Training objective. The base model θ_0 is frozen and only the low-rank adaptation (LoRA) parameters $\Delta\theta$ are trained. With x the serialized instruction and context and $y = (y_1, \ldots, y_m)$ the short answer, client k minimizes the cross-entropy of the answer tokens,

$$\mathcal{L}_k(\Delta\theta; x, y) = -\sum_{i=1}^{m} \log p_{\theta_0+\Delta\theta}(y_i \mid x, y_{<i}), \tag{6}$$

with all tokens of x masked out of the loss.

Stage order. Training proceeds through the three stages in a fixed order, write, then recall, then protect, in contiguous blocks of 6, 6, and 5 federated rounds for $T = 17$. Recall supervision is meaningful only after the write stage has associated the question, the omitted evidence, and the answer, and protection comes last so that no later stage can overwrite its effect on retrieval hits. Recall and protection pull the adapter in two directions, and Section 4.3 measures the cost of the order.

3.3 Federated adapter aggregation

Federated adapter aggregation runs the three stages in federated rounds and closes every round by averaging the client adapters, weighted by the examples each visited. At round t the server broadcasts the current global adapter $\Delta\theta^t$, each client performs a fixed number of local optimizer updates on the current stage's task set $\mathcal{T}_{k,t}$ starting from it, $\Delta\theta_k^{t+1} = \mathrm{LocalUpdate}(\Delta\theta^t, \mathcal{T}_{k,t})$, and uploads the adapter alone. The server averages the client adapters, weighting each client by the number $n_{k,t}$ of examples it actually visited during round t,

$$\Delta\theta^{t+1} = \sum_k \frac{n_{k,t}}{\sum_j n_{j,t}} \Delta\theta_k^{t+1}. \tag{7}$$

The weights reduce to uniform weighting when every client fills its fixed update budget.

Equation (7) is standard federated averaging. What is specific to FedR^{2}M is what the average carries. Passages, questions, and retrieval residual labels stay with the client that owns them, so the global adapter is the only channel through which a retrieval residual observed at one client can change how another client's questions are answered. A local-only control matched on total optimization tests whether that channel carries useful knowledge.

Communication cost. With C clients, T rounds, and an adapter payload of S bytes, each client downloads and uploads the adapter once per round, so the training communication is $B_{\mathrm{comm}} = 2\,C\,T\,S$, linear in C, and we report quality and communication together whenever the client count varies.

3.4 Global adapter inference

Global adapter inference answers every question with the unchanged retriever and the one global adapter. After the final protection round, the global adapter $\Delta\theta^\star$ is frozen and shipped to every client, and for a question q against the serving client's deployed corpus $\mathcal{D}$,

$$\hat{a}_q \sim p_{\theta_0+\Delta\theta^\star}(a \mid q,\, R_K(q, \mathcal{D})). \tag{8}$$

There is no adapter bank, no per-question adapter selection, no gating network, and no routing at inference time. The ordered top-K, the prompt, the decoding settings, and the scorer are those of retrieval-only prompting, so any difference in behavior is attributable to the adapter alone.

4 Experiments

4.1 Experimental setup

We use four open-domain question-answering benchmarks, HotpotQA (Yang et al., 2018) and 2WikiMultiHopQA (2Wiki) (Ho et al., 2020) with annotated supporting facts, PopQA (Mallen et al., 2023) with long-tail entities, and ComplexWebQuestions (CWQ) (Talmor & Berant, 2018) with compositional questions, and keep four evaluation sets apart. The main comparison holds out 300

Table 1: Main comparison on four QA benchmarks. Bold marks the best compared system.

	F1 by benchmark				Macro		F1 by retrieval label	
Method	HotpotQA	2Wiki	PopQA	CWQ	F1	EM	Hits	Misses
No shared corpus								
Local model	0.1078	0.4128	0.0920	0.1902	0.2007	0.1592	0.2598	0.1402
Retrieved text								
MKPQA	0.2337	0.3750	0.2945	0.2597	0.2907	0.2108	0.4340	0.1441
Dense retrieval	0.2193	0.3521	0.3002	0.2693	0.2852	0.2050	0.4464	0.1202
Retrieval-only prompting	0.2269	0.3832	0.3133	0.2703	0.2984	0.2158	0.4600	0.1330
Learned routing	0.2269	0.3832	0.3136	0.2703	0.2985	0.2158	0.4601	0.1330
RAGRouter	0.2340	0.3843	0.3047	0.2629	0.2965	0.2158	0.4578	0.1313
Document adapters								
FedMosaic	0.1389	0.4172	0.2411	0.3023	0.2749	0.2225	0.4109	0.1357
DistilledPRAG	0.1180	0.4391	0.1371	0.3971	0.2728	0.2242	0.3282	**0.2161**
FedR^{2}M	**0.2989**	**0.4455**	**0.4288**	**0.4565**	**0.4074**	**0.3483**	**0.5990**	0.2114
Reference								
Centralized	0.3025	0.4419	0.4180	0.4592	0.4054	0.3542	0.6057	0.2004

questions per benchmark, 607 retrieval hits and 593 retrieval misses by the alias test of Appendix A. A holdout of 400 training-side HotpotQA and 2Wiki questions serves the stage and aggregation analyses, a blind test set of 1,000 questions from the same two benchmarks, whose answers were revealed only after every prediction had been hashed, serves the mechanism test, the same-question comparison uses the 1,200 questions and prebuilt top-3 passages of the official Parametric RAG protocol, and the per-benchmark analysis reuses the 1,200 questions of the main comparison under the four settings of the blind test, and those questions were in use during development and are therefore not a blind test set. The main comparison, the holdout, and the same-question comparison use Llama-3.2-1B-Instruct, the blind test and the four-benchmark set use the 8B adapter of centralized training with Llama-3.1-8B-Instruct (Grattafiori et al., 2024), the retriever is BM25 (Robertson & Zaragoza, 2009) with a fixed top-5, and answers are scored by normalized token F1 and exact match with the maximum over aliases. Six clients train LoRA adapters for 17 rounds, 612 updates, and 19,584 example visits, centralized training of the same memory uses 100 updates on one machine, every table other than the seed study reports the run at seed 1729, and differences come with 95% paired bootstrap confidence intervals. Table 1 groups the compared systems by whether they use no shared corpus, read retrieved text, or carry the shared documents as adapter parameters, with centralized training, the Centralized row, as the reference rather than a competing system, and Appendix A describes every system, the controls, and the training configuration.

4.2 Comparison with baselines

Main comparison. FedR^{2}M outperforms both the systems that share document adapters and the systems that read retrieved text, and it matches centralized training of the same memory. In Table 1 it improves macro F1 by at least 36% over every baseline, by 37% over retrieval-only prompting and by 48% over FedMosaic, which scores highest among the document-adapter systems, with $p{=}0.0002$ for every margin, and the margins hold on HotpotQA, PopQA, and CWQ, while DistilledPRAG is within 2% on 2Wiki. The document-adapter systems gain their memory at the cost of the retrieval hits, where both score below retrieval-only prompting, whereas FedR^{2}M improves on retrieval-only prompting by 30% on the hits and by 59% on the misses. Federation shows no measurable loss against centralized training, with a paired difference of 0.0020 and a 95% CI of $[-0.0161, 0.0197]$. At 8B, FedR^{2}M improves on every baseline run at that scale by at least 48%, nearly triples miss F1 over retrieval-only prompting, from 0.1231 to 0.3507, and is 2% below centralized training with a CI that contains zero.

Same-question comparison with Parametric RAG. One frozen global adapter outperforms per-passage document parameterization on the same inputs with about a thousandth of the storage. In Table 2, it improves F1 by 41% over Parametric RAG, with a 95% CI of $[0.0799, 0.1303]$, and by 39% over no-adapter prompting, storing 13.02 MiB in place of 13.23 GiB of per-passage adapters,

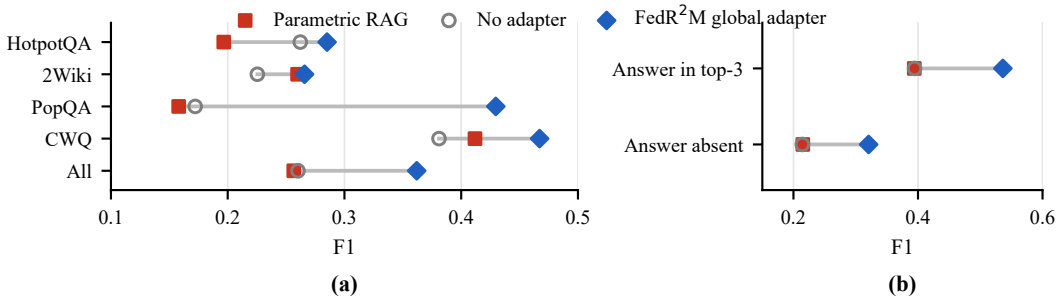

Figure 2: Same-question comparison with Parametric RAG. (a) F1 by benchmark. (b) Split by whether the answer appears in the shared top-3.

Table 2: Same-question comparison with Parametric RAG.

Method	F1	Updates	Adapters	Storage	Training
Parametric RAG, per passage	0.2566	22,500	3,600	13.23 GiB	8.92 h
No adapter	0.2602	0	0	0	0
FedR^{2}M global adapter	**0.3619**	612	1	13.02 MiB	0.41 h

although Parametric RAG trains on the passages of these very questions. In Figure 2, the margin holds on all four benchmarks, by 36% where the answer is in the shared top-3 and by 49% where it is not.

4.3 Ablation Study

Boundary and pairing. Selecting what to memorize by retrieval failure outperforms memorizing everything, and pairing each question with its own evidence and answer outperforms spending the same budget on permuted ones. In the blind test of Table 3 and Figure 3, retrieval residual memory improves F1 by 13% over the no-boundary control, with a 95% CI of $[0.0537, 0.1028]$, and by 77% on the retrieval residuals over the permuted control, with a 95% CI of $[0.0985, 0.2136]$. Retrieval residual memory also improves the retrieval hits by 23% over retrieval-only prompting and scores highest on every label and both benchmarks. A budget-matched no-boundary control that spends the identical budget and pipeline on 1,600 questions selected without the retrieval-failure filter, 860 of them retrieval hits, still scores 4% below retrieval residual memory on the blind test set, with a 95% CI of $[0.0079, 0.0450]$, and 14% below on the retrieval residuals, so the boundary's gain comes from directing the same budget at what retrieval misses.

Training stages. The recall stage raises the retrieval residuals and lowers the retrieval hits, and the protection stage recovers the hits while keeping most of recall's gain. On the holdout in Figure 4(a), recall lifts F1 on the retrieval residuals by 40% and lowers it on the hits by 19%. Protection then raises the hits by 66%, lifts the whole holdout by 35% to 0.6381, and costs 11% on the residuals. Placing recall after protection instead undoes what protection recovered, holdout hits fall from 0.7241 to 0.4909, and the fixed order improves holdout F1 by 27% over that alternative, with a 95% CI of $[0.0929, 0.1761]$.

Federated aggregation. Aggregation, not longer training, produces the gain from federation. In Table 4, the local-only control spends the same 612 updates and 19,584 example visits but trains each client alone, and the full protocol improves F1 over it by 8%, with a 95% CI of $[0.0108, 0.0875]$ and $p = 0.0114$, so knowledge learned from retrieval residuals at one client reaches the others through the global adapter. Every client benefits, answering its own holdout questions better with the global adapter than with retrieval-only prompting, by 8% to 88% in Figure 4(b).

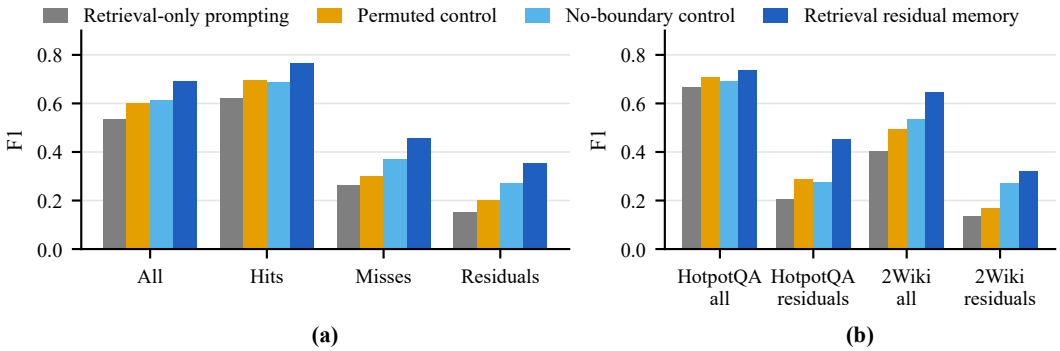

Figure 3: Blind mechanism test. (a) F1 by retrieval label. (b) F1 by benchmark.

Table 3: Blind mechanism test, split by retrieval label.

Method	All	Hits	Misses	Residuals
Retrieval-only prompting	0.5347	0.6200	0.2619	0.1525
Permuted control	0.6007	0.6949	0.2993	0.1999
No-boundary control	0.6112	0.6862	0.3711	0.2706
Retrieval residual memory	**0.6894**	**0.7630**	**0.4538**	**0.3546**

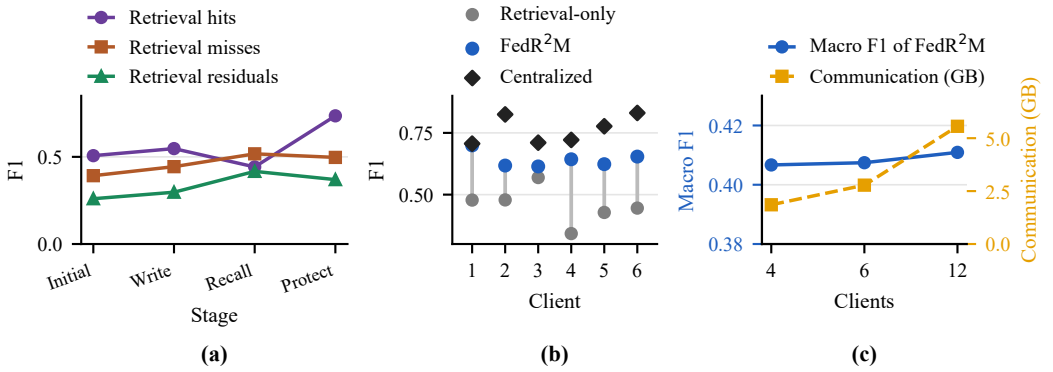

Figure 4: (a) Holdout F1 by retrieval label after each training stage. (b) Per-client holdout F1 with retrieval-only prompting, with FedR^{2}M, and with centralized training. (c) Macro F1 and communication cost at 4, 6, and 12 clients.

Table 4: Aggregation control on the holdout set.

Training	F1	Updates	Example visits
Retrieval-only prompting	0.4601	0	0
Local-only control	0.5894	612	19,584
FedR^{2}M	**0.6381**	612	19,584

4.4 Scalability and Cost

Client count has little effect on quality and multiplies communication. Across 4, 6, and 12 clients at the same total number of updates and example visits in Figure 4(c), macro F1 varies by 1% while communication triples from 1.85 GB to 5.56 GB. The 2.78 GB that six clients exchange over the 17 rounds is under one percent of what exchanging the full 1B model over the same rounds would move and less than the 4.7 GiB of document adapters and masks that FedMosaic shares for the same four benchmarks.

Federation costs training time and the adapter costs latency. On one A6000 GPU the six-client protocol trains in 1,473.01 s, 5.8 times the 253.13 s of centralized training, and the global adapter adds about 0.42 s to each 64-token answer, from 776.87 ms to 1,192.75 ms, with the retrieval payload unchanged.

One global adapter needs less offline training and less storage than the document-adapter systems, and no adapter built per question. In Table 5, FedMosaic spends 19.9 GPU-hours of offline training on the four benchmarks against 0.41 for the six-client protocol, 49 times more, stores 4.7 GiB of document adapters and masks for the four benchmarks against one 13 MiB adapter, 369 times more, and, without an adapter cache, attaches 7.4 to 8.0 MiB of adapters and masks to every question. DistilledPRAG trains in under an hour, but it carries a 321 MiB generator and builds an adapter for every question at query time.

Table 5: Offline cost of the document-adapter systems.

Method	Offline training (s)	Stored at inference	Adapter bytes per question
FedMosaic	71,646	4,801.3 MiB	7.37 to 8.01 MiB
DistilledPRAG	2,697	321.3 MiB	generated per question
FedR^{2}M	1,473	13.02 MiB	0

4.5 Per-benchmark analysis

The blind-test conclusions replicate on the four-benchmark set at 8B, with one exception on 2Wiki. Retrieval residual memory improves macro F1 by 13% over the no-boundary control, with a 95% CI of $[0.0341, 0.0822]$, and by 50% over retrieval-only prompting, and it scores highest on three of the four benchmarks. On 2Wiki it still improves over retrieval-only prompting by 21% but is 7% below the no-boundary control, and the gap is confined to the 42 Boolean comparison questions, 33 of which have yes as the reference answer, where the no-boundary control answers yes on 41 and benefits from a nearly constant answer that matches the label imbalance, while retrieval residual memory is 4% above it on the remaining 258 questions.

5 Conclusion

FedR^{2}M treats the persistent misses of a fixed retriever as the criterion for what a language model should carry in its parameters. Across four benchmarks the global adapter outperforms every compared system on macro F1, matches centralized training of the same memory without pooling raw data, and outperforms local training matched on total optimization, and a blind test attributes the gain to both the boundary and the pairing. Managing the interference between new memory and reliable retrieval more fully is the next step. Retrieval failures are not only a defect to repair but an auditable boundary around the knowledge that belongs in parameters.

Limitations. The gains have a measured price. Keeping raw data local costs training time and communication, six-client training taking about 5.8 times the wall-clock time of centralized training and 2.78 GB of communication that grows linearly with the number of clients, and the global adapter adds about 0.42 s to each answer. The privacy property this work establishes is data locality. Stronger requirements can be met with secure aggregation or differentially private training on the adapter exchange and with protected retrieval for the passages that the ordinary inference path sends in plaintext, and these additions are orthogonal to the method.

AI use statement

Large language model based tools were used to aid and polish the writing of this paper, including language editing and checks of the consistency between the text, the tables, and the figures. The authors reviewed all of the text and take full responsibility for its content.

REFERENCES

Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of ACM CCS*, pp. 308–318, 2016. doi: 10.1145/2976749.2978318.

Zeyuan Allen-Zhu and Yuanzhi Li. Physics of language models: Part 3.1, knowledge storage and extraction. In *Proceedings of ICML*, volume 235, pp. 1067–1077, 2024.

Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi. Self-RAG: Learning to retrieve, generate, and critique through self-reflection. In *International Conference on Learning Representations*, 2024.

Seungju Back, Dongwoo Lee, Naun Kang, Taehee Lee, S. K. Hong, Youngjune Gwon, and Sungjin Ahn. Understanding LoRA as knowledge memory: An empirical analysis. In *Proceedings of ICML*, 2026.

Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan McMahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. In *Proceedings of ACM CCS*, pp. 1175–1191, 2017. doi: 10.1145/3133956.3133982.

Jinwen Chen, Hainan Zhang, Liang Pang, Yongxin Tong, Haibo Zhou, Wei Lin, and Zhiming Zheng. Privacy-preserving reasoning with knowledge-distilled parametric retrieval augmented generation. In *Findings of the Association for Computational Linguistics: ACL 2026*, 2026.

Val Andrei Fajardo, David B. Emerson, Amandeep Singh, Veronica Chatrath, Marcelo Lotif, Ravi Theja, Alex Cheung, and Izuki Matsuba. FedRAG: A framework for fine-tuning retrieval-augmented generation systems. In *ICML Workshop on Championing Open-source Development in Machine Learning*, 2025.

William Fleshman and Benjamin Van Durme. LoRA-augmented generation (LAG) for knowledge-intensive language tasks. *arXiv preprint arXiv:2507.05346*, 2025. URL https://arxiv.org/abs/2507.05346.

Tianhao Gao, Kai Yang, and Yiyang Li. FD-RAG: Federated dual-system retrieval-augmented generation. *arXiv preprint arXiv:2605.27432*, 2026. URL https://arxiv.org/abs/2605.27432.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Rachid Guerraoui, Anne-Marie Kermarrec, Diana Petrescu, Rafael Pires, Mathis Randl, and Martijn de Vos. Efficient federated search for retrieval-augmented generation. In *Proceedings of the 5th Workshop on Machine Learning and Systems (EuroMLSys)*, pp. 74–81. Association for Computing Machinery, 2025. doi: 10.1145/3721146.3721942.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training. In *Proceedings of ICML*, volume 119, pp. 3929–3938, 2020.

Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa. Constructing a multi-hop QA dataset for comprehensive evaluation of reasoning steps. In *Proceedings of COLING*, pp. 6609–6625, 2020. doi: 10.18653/v1/2020.coling-main.580.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Atlas: Few-shot learning with retrieval augmented language models. *Journal of Machine Learning Research*, 24(251):1–43, 2023.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *International Conference on Learning Representations*, 2022.

Fengxian Ji, Zhuohan Xie, Jingpu Yang, Fan Zhang, Zirui Song, and Xiuying Chen. Parametric memory decoding for zero-shot routing in LoRA-based external parametric memory. *arXiv preprint arXiv:2607.04118*, 2026. URL https://arxiv.org/abs/2607.04118.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of EMNLP*, pp. 7969–7992, 2023. doi: 10.18653/v1/2023.emnlp-main.495.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of EMNLP*, pp. 6769–6781, 2020. doi: 10.18653/v1/2020.emnlp-main.550.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Zhilin Liang, Yuxiang Wang, Zimu Zhou, Hainan Zhang, Boyi Liu, and Yongxin Tong. FedMosaic: Federated retrieval-augmented generation via parametric adapters. *arXiv preprint arXiv:2602.05235*, 2026. URL https://arxiv.org/abs/2602.05235.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. Entity-based knowledge conflicts in question answering. In *Proceedings of EMNLP*, pp. 7052–7063, 2021. doi: 10.18653/v1/2021.emnlp-main.565.

Xinbei Ma, Yeyun Gong, Pengcheng He, Hai Zhao, and Nan Duan. Query rewriting in retrieval-augmented large language models. In *Proceedings of EMNLP*, pp. 5303–5315, 2023. doi: 10.18653/v1/2023.emnlp-main.322.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of ACL*, pp. 9802–9822, 2023.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Proceedings of AISTATS*, volume 54, pp. 1273–1282, 2017.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. Locating and editing factual associations in GPT. In *Advances in Neural Information Processing Systems*, volume 35, pp. 17359–17372, 2022.

Julian Oestreich, Maximilian Bley, Frank Binder, Lydia Müller, André Alcalde, and Maksym Sydorenko. Parametric knowledge in RAG-SFT for domain-specific document generation. *arXiv preprint arXiv:2603.23047*, 2026. URL https://arxiv.org/abs/2603.23047.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. Fine-tuning or retrieval? Comparing knowledge injection in LLMs. In *Proceedings of EMNLP*, pp. 237–250, 2024. doi: 10.18653/v1/2024.emnlp-main.15.

Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009. doi: 10.1561/1500000019.

David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. In *Advances in Neural Information Processing Systems*, volume 32, pp. 350–360, 2019.

Weijia Shi, Xiaochuang Han, Mike Lewis, Yulia Tsvetkov, Luke Zettlemoyer, and Wen-tau Yih. Trusting your evidence: Hallucinate less with context-aware decoding. In *Proceedings of NAACL-HLT*, pp. 783–791, 2024a. doi: 10.18653/v1/2024.naacl-short.69.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. REPLUG: Retrieval-augmented black-box language models. In *Proceedings of NAACL-HLT*, pp. 8371–8384, 2024b. doi: 10.18653/v1/2024.naacl-long.463.

Parshin Shojaee, Sai Sree Harsha, Dan Luo, Akash Maharaj, Tong Yu, and Yunyao Li. Federated retrieval augmented generation for multi-product question answering. In *Proceedings of the 31st International Conference on Computational Linguistics: Industry Track*, pp. 387–397. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.coling-industry.33/.

Weihang Su, Yichen Tang, Qingyao Ai, Junxi Yan, Changyue Wang, Hongning Wang, Ziyi Ye, Yujia Zhou, and Yiqun Liu. Parametric retrieval augmented generation. In *Proceedings of SIGIR*, 2025a. doi: 10.1145/3726302.3729957.

Zhan Su, Fengran Mo, Jinghan Zhang, Yuchen Hui, Jiaao Sun, and Jian-yun Nie. Parametric retrieval-augmented generation using latent routing of LoRA adapters. *arXiv preprint arXiv:2511.17044*, 2025b. URL https://arxiv.org/abs/2511.17044.

Youbang Sun, Zitao Li, Yaliang Li, and Bolin Ding. Improving LoRA in privacy-preserving federated learning. In *International Conference on Learning Representations*, 2024.

Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. In *Proceedings of NAACL-HLT*, pp. 641–651, 2018. doi: 10.18653/v1/N18-1059.

Yuqiao Tan, Shizhu He, Huanxuan Liao, Jun Zhao, and Kang Liu. Dynamic parametric retrieval augmented generation for test-time knowledge enhancement. *arXiv preprint arXiv:2503.23895*, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Interleaving retrieval with chain-of-thought reasoning for knowledge-intensive multi-step questions. In *Proceedings of ACL*, pp. 10014–10037, 2023. doi: 10.18653/v1/2023.acl-long.557.

Jian Xie, Kai Zhang, Jiangjie Chen, Renze Lou, and Yu Su. Adaptive chameleon or stubborn sloth: Revealing the behavior of large language models in knowledge conflicts. In *International Conference on Learning Representations*, 2024.

Lei Xu, Xin Quan, Daniel Pedronette, and André Freitas. AbstRAG: Learning to abstract for retrieval problems. *arXiv preprint arXiv:2606.09459*, 2026. URL https://arxiv.org/abs/2606.09459.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of EMNLP*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Jiarui Zhang, Xiangyu Liu, Yong Hu, Chaoyue Niu, Fan Wu, and Guihai Chen. RAGRouter: Learning to route queries to multiple retrieval-augmented language models. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Zhuo Zhang, Yuanhang Yang, Yong Dai, Qifan Wang, Yue Yu, Lizhen Qu, and Zenglin Xu. FedPETuning: When federated learning meets the parameter-efficient tuning methods of pre-trained language models. In *Findings of ACL*, pp. 9963–9977, 2023. doi: 10.18653/v1/2023.findings-acl.632.

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

Zhengyi Zhao, Shubo Zhang, Zezhong Wang, Yuxi Zhang, Huimin Wang, Yutian Zhao, Yefeng Zheng, Binyang Li, Kam-Fai Wong, and Xian Wu. Guaranteeing knowledge integration with joint decoding for retrieval-augmented generation. In *Proceedings of ACL*, pp. 4181–4205, 2026. doi: 10.18653/v1/2026.acl-long.192.

Shuaipeng Zhou and Yu Zhang. SCALE-LoRA: Auditing post-retrieval LoRA composition with residual merging and view reliability. *arXiv preprint arXiv:2605.01429*, 2026. URL https://arxiv.org/abs/2605.01429.

Ligeng Zhu, Zhijian Liu, and Song Han. Deep leakage from gradients. In *Advances in Neural Information Processing Systems*, volume 32, pp. 14774–14784, 2019.

Chunsheng Zuo, Liaoyaqi Wang, William Jurayj, William Fleshman, and Benjamin Van Durme. Rethinking LoRA memory through the lens of KV cache compression. *arXiv preprint arXiv:2606.05698*, 2026. URL https://arxiv.org/abs/2606.05698.

A Evaluation sets and training details

Evaluation sets. The main comparison holds out 300 questions per benchmark, 1,200 in total. The holdout used for the stage and aggregation analyses contains 400 training-side questions, 200 each from HotpotQA and 2Wiki, partitioned across the six clients, and each client's local-only adapter is evaluated on that client's own questions. The blind test set contains 1,000 questions, 500 each from HotpotQA and 2Wiki, drawn from the official validation splits after excluding every question that had appeared in an earlier evaluation, so none of them had been used for task construction or model selection, and its reference answers were revealed only after every prediction file had been hashed. By the alias test it splits into 762 retrieval hits and 238 retrieval misses, 173 of which are retrieval residuals under the string-matching test. The four-benchmark set is the 1,200 questions of the main comparison, evaluated with the 8B settings of the blind test. The same-question comparison uses the 1,200 questions and the prebuilt ordered top-3 passages of the official Parametric RAG implementation.

Data construction. Retrieval residual discovery uses the official training splits of HotpotQA and 2WikiMultiHopQA, hashed by question identifier into ten disjoint folds, of which folds 0 to 7 construct the training examples, fold 9 is the 400-question holdout, and fold 8 is set aside. Each question carries the ten candidate paragraphs of its benchmark release, the fixed retriever ranks them by BM25 over title and text with $k_1 = 1.5$ and $b = 0.75$ and returns five, and local availability is checked against the same ten candidates, which stay with the question on its client. Construction takes, per benchmark and in a fixed hash order, the first 800 retrieval residual questions and the first 600 retrieval hits by the alias test, giving 1,600 write, 1,600 recall, and 1,200 protection examples. The six clients hold disjoint question sets, three per benchmark, a question going to the client that owns its primary supporting document under a hash of the document title, so every client holds between 457 and 480 training questions. Training sequences are capped at 512 tokens and truncated from the front of the prompt, so the missing evidence, the question, and the answer at the end of the prompt are retained in full in every example. The main comparison uses the 300 questions per benchmark of the Parametric RAG evaluation release in the official FedMosaic split, each question with its released corpus of 1,500 candidate documents distributed over six silos, and the per-benchmark analysis reuses these questions, so the main comparison and the per-benchmark analysis run under a retriever and corpus other than the ones the residuals were discovered on. The shared BM25 top-5 ranks all 1,500 documents, the local model sees one silo, the routing systems choose among silos, and hits and misses on all four benchmarks come from the alias test over the shared top-5.

String-matching test of the labels. Every label is computed from the frozen retriever output and the client's own annotations before any adapter is trained. The alias test marks a question as a retrieval hit when a normalized alias of its reference answer occurs in the fixed top-K, and every hit and miss count in the paper uses it. The retrieval residual boundary h_q of Equation 2 also counts a question as covered when the top-K holds every passage carrying its annotated supporting facts, and in the labels of the construction pool, the holdout, and the blind test set "yes" and "no" are not counted as aliases, since they occur in passages without being evidence. Local availability is checked against the

Algorithm 1 FedR^{2}M

Require: frozen retriever R_K; clients $k = 1, \ldots, C$ with corpora $\mathcal{D}_k$ and questions $\mathcal{Q}_k$; frozen base model θ_0; the stage schedule of write, recall, and protection rounds, T rounds in total
Ensure: one global adapter $\Delta\theta^\star$, served with the unchanged retriever for every question

for each client k, locally **do**
  retrieve $\mathcal{C}_q = R_K(q, \mathcal{D}_k)$ and compute h_q, v_q, and $z_q = v_q(1 - h_q)$ for every $q \in \mathcal{Q}_k$
  build write and recall examples from the questions with $z_q = 1$ and protection examples from the retrieval hits of the alias test
end for
for round $t = 0, \ldots, T-1$, in the stage order write, recall, protect **do**
  the server broadcasts $\Delta\theta^t$
  for each client k, in parallel **do**
    starting from $\Delta\theta^t$, spend the fixed local update budget on the current stage's task set $\mathcal{T}_{k,t}$ with Equation (6), and upload the result $\Delta\theta_k^{t+1}$
  end for
  $\Delta\theta^{t+1} \leftarrow \sum_k \frac{n_{k,t}}{\sum_j n_{j,t}} \Delta\theta_k^{t+1}$, with $n_{k,t}$ the examples client k visited in round t
end for
return $\Delta\theta^\star = \Delta\theta^T$

client's complete candidate pool, so v_q measures corpus coverage and h_q retriever coverage, and the test needs no communication and never consults evaluation outcomes.

Training and decoding. Clients train LoRA on the query, key, value, and output projections with rank 16, scaling 32, and dropout 0.05, in bfloat16 with a maximum sequence length of 512, micro-batches of 2 with 16 accumulation steps (effective batch 32), AdamW with learning rate 2×10^{-4}, weight decay 0.01, gradient clipping at 1.0, and seed 1729, with seeds 1730 and 1731 added for the seed study. The six-client protocol runs 17 rounds with 6 local updates per client per round and a fresh optimizer state at every round, 6 write, 6 recall, and 5 protection rounds, for 612 updates and 19,584 example visits, and the 4- and 12-client settings use 9 and 3 local updates per round so that both totals stay fixed. Centralized training uses the same examples on one machine with 100 updates and 3,200 example visits. The single-machine 8B adapters of the blind test and the four-benchmark set use 35 write, 35 recall, and 30 protection updates, a linear schedule, and 3% warmup, and decode greedily with at most 32 new tokens in a 4,096-token input window; the 8B federated run of Table 9 keeps the six-client schedule and the decoding of the main comparison. Table 6 summarizes these settings. Paired bootstrap intervals use 10,000 replicates on the main comparison, the blind test set, and the holdout, and 2,000 on the same-question comparison.

Table 6: Training and decoding configuration.

Component	Value
Main comparison model	Llama-3.2-1B-Instruct
Mechanism test model	Llama-3.1-8B-Instruct
Retriever	BM25, ordered top-5 (top-3 in the same-question comparison)
LoRA	rank 16, scaling 32, dropout 0.05, on query, key, value, and output projections
Optimizer	AdamW, learning rate 2×10^{-4}, weight decay 0.01, clipping 1.0
Effective batch	32 (micro-batch 2, accumulation 16), bfloat16, sequence length 512
Federation	6 clients, 17 rounds, 6 local updates per round, 612 updates, 19,584 visits
Stage rounds	write 6, recall 6, protect 5
Single-machine schedule	100 updates (write 35, recall 35, protect 30), linear schedule, 3% warmup
Seed	1729 (every reported run and the bootstrap); 1730 and 1731 only in the seed study of Table 7
Blind-test decoding	greedy, 4,096-token input, at most 32 new tokens
Scorer	normalized token F1 and exact match, maximum over answer aliases

Compared systems. Every system in the main comparison keeps its own retrieval or memory protocol under the same base model, question set, prompt, and scorer. The local model answers from

the client's own documents alone. Five systems read text retrieved from the shared corpus, retrieval-only prompting with the shared BM25 top-5, dense retrieval with the retriever of FRAG (Zhao, 2024), and three routing systems over the six silos, MKPQA (Shojaee et al., 2025), learned routing (Guerraoui et al., 2025), and RAGRouter (Zhang et al., 2025), the last three our reimplementations. Two systems carry the shared documents as adapter parameters, FedMosaic (Liang et al., 2026) and DistilledPRAG (Chen et al., 2026), both with their official implementations. Centralized training pools the six clients' examples and trains the same memory as FedR^{2}M on one machine, the 8B federated run keeps the six-client schedule, the blind test and the four-benchmark set use the 8B adapter of centralized training, centralized training and FedR^{2}M are retrained with two further seeds, and timing is measured on one A6000 GPU. The same-question comparison reproduces the official Parametric RAG implementation (Su et al., 2025a), one adapter per passage of the shared top-3, merged per question at test time.

Controls. Four controls isolate the boundary, the pairing, and aggregation. The permuted control keeps the questions, retrieved contexts, example and update counts, and hyperparameters of the corresponding FedR^{2}M run and gives each question the missing evidence and answer of another question of the same benchmark and question type in the write and recall stages. The no-boundary control memorizes the corpus without the retrieval residual boundary under its own budget. The local-only control gives each client its own adapter and spreads the full protocol's updates and example visits across the clients without aggregation. The budget-matched no-boundary control of Appendix B keeps the budget and pipeline of retrieval residual memory and removes only the boundary.

Baseline implementations. MKPQA reimplements MKP-QA with a classifier over a query encoder that selects the silos and fuses their retrieval scores. Learned routing reimplements the RAGRoute router over the six silos. RAGRouter has no released router, so it is implemented from its paper, each query going to the silo whose local BM25 top passage scores highest, and that best-silo strategy is the reported row. DistilledPRAG uses the official shared generator, which produces one adapter from the retrieved group of each question, with the same frozen base model as teacher and student. FedMosaic and Parametric RAG run their official implementations.

B Additional results

Seed variance and exact match on the main comparison. Table 7 reports centralized training and FedR^{2}M from Table 1 at three training seeds with the mean and sample standard deviation, and Table 8 gives exact match by benchmark for every method. The seed-1729 adapters are the ones used in every other experiment of the paper.

Table 7: Seed variance of centralized training and FedR^{2}M on the main comparison.

Method	Seed	HotpotQA	2Wiki	PopQA	CWQ	Macro F1	Macro EM	Hits	Misses
FedR^{2}M	1729	0.2989	0.4455	0.4288	0.4565	0.4074	0.3483	0.5990	0.2114
	1730	0.2995	0.4379	0.4407	0.4390	0.4043	0.3367	0.5928	0.2113
	1731	0.3016	0.4613	0.4344	0.4486	0.4115	0.3517	0.5906	0.2281
	Mean	0.3000	0.4482	0.4346	0.4480	0.4077	0.3456	0.5941	0.2169
	Std	0.0014	0.0120	0.0059	0.0088	0.0036	0.0079	0.0043	0.0097
Centralized	1729	0.3025	0.4419	0.4180	0.4592	0.4054	0.3542	0.6057	0.2004
	1730	0.3032	0.4356	0.4306	0.4662	0.4089	0.3542	0.6123	0.2007
	1731	0.2993	0.4028	0.4178	0.4664	0.3966	0.3425	0.6030	0.1852
	Mean	0.3017	0.4268	0.4221	0.4639	0.4036	0.3503	0.6070	0.1955
	Std	0.0021	0.0210	0.0073	0.0041	0.0064	0.0067	0.0047	0.0089

The main comparison with the 8B model. Table 9 repeats the main comparison with Llama-3.1-8B-Instruct on the same 1,200 questions, BM25 top-5, prompt, decoding, and scorer, for the methods whose protocol transfers to the larger model without refitting. FedR^{2}M is trained at 8B under the six-client schedule of Appendix A with seed 1729, and the centralized training row is the 8B single-machine adapter of the blind test, evaluated on these questions. Paired bootstraps with 10,000 replicates give FedR^{2}M a margin of 0.1689 over retrieval-only prompting, 95% CI $[0.1423, 0.1963]$, then 0.1591 over dense retrieval and 0.3058 over the local model, each with $p = 0.0002$. Against

Table 8: Exact match on the main comparison.

Method	HotpotQA	2Wiki	PopQA	CWQ	Macro
No shared corpus					
Local model	0.0667	0.3933	0.0533	0.1233	0.1592
Retrieved text					
MKPQA	0.1500	0.3333	0.1800	0.1800	0.2108
Dense retrieval	0.1567	0.3100	0.1767	0.1767	0.2050
Retrieval-only prompting	0.1600	0.3400	0.1867	0.1767	0.2158
Learned routing	0.1600	0.3400	0.1867	0.1767	0.2158
RAGRouter	0.1700	0.3400	0.1767	0.1767	0.2158
Document adapters					
FedMosaic	0.0933	0.3667	0.2100	0.2200	0.2225
DistilledPRAG	0.0633	0.4233	0.1133	0.2967	0.2242
FedR^{2}M	**0.2133**	**0.4400**	**0.3667**	**0.3733**	**0.3483**
Reference					
Centralized	0.2233	0.4233	0.3867	0.3833	0.3542

centralized training the pooled difference is -0.0117 with a 95% CI of $[-0.0337, 0.0105]$ and $p = 0.2906$. As at 1B, centralized training is higher on the retrieval hits and FedR^{2}M on the retrieval misses.

Table 9: Main comparison with the 8B model.

	F1 by benchmark				Macro		F1 by retrieval label	
Method	HotpotQA	2Wiki	PopQA	CWQ	F1	EM	Hits	Misses
No shared corpus								
Local model	0.0999	0.3006	0.1078	0.2129	0.1803	0.0967	0.2655	0.0931
Retrieved text								
Retrieval-only prompting	0.2932	0.3780	0.3029	0.2948	0.3172	0.1833	0.5068	0.1231
Dense retrieval	0.3001	0.3498	0.2999	0.3582	0.3270	0.1958	0.5094	0.1403
FedR^{2}M	**0.4625**	**0.4998**	**0.4619**	**0.5204**	**0.4861**	**0.4183**	**0.6185**	**0.3507**
Reference								
Centralized	0.4747	0.5067	0.4508	0.5592	0.4978	0.4292	0.6930	0.2980

Blind test set by benchmark and label. Table 10 gives the values behind Figure 3 for both benchmarks of the blind test set.

Table 10: F1 on the blind test set by benchmark and retrieval label.

	HotpotQA			2Wiki		
Method	All	Hits	Residuals	All	Hits	Residuals
n	500	431	46	500	331	127
Retrieval-only prompting	0.6674	0.7125	0.2029	0.4021	0.4995	0.1342
Permuted control	0.7065	0.7425	0.2881	0.4949	0.6328	0.1679
No-boundary control	0.6902	0.7322	0.2754	0.5323	0.6264	0.2689
Retrieval residual memory	**0.7349**	**0.7650**	**0.4512**	**0.6440**	**0.7605**	**0.3196**

Budget-matched no-boundary control. A no-boundary control matched to retrieval residual memory in budget and pipeline selects 1,600 questions from the same construction pool without the retrieval-failure filter, stratified by benchmark, question type, and Boolean answers, of which 860 are retrieval hits and 607 retrieval residuals, keeps the same 1,200-question protection pool and the same

35, 35, and 30 updates, and was evaluated after the withheld answers had been revealed, so it is not a blind prediction. In one evaluation pass over the four settings, the matched control scores 0.0261 below retrieval residual memory on the blind test set, with a 95% CI of $[0.0079, 0.0450]$, 0.0492 below on the retrieval residuals and 0.0610 below on the retrieval misses, and 0.0518 above the no-boundary control of Table 3. On the four-benchmark set it scores 0.0168 below retrieval residual memory, with a 95% CI of $[-0.0019, 0.0353]$, and below it on all four benchmarks.

Per-client gains on the holdout. Table 11 gives the values behind Figure 4(b).

Table 11: Per-client holdout F1 before and after each training stage.

Client	n	Initial	Write	Recall	Protect (final)	Centralized	Gain
1	52	0.4779	0.5173	0.4865	0.6994	0.7064	$+0.2214$
2	82	0.4786	0.5173	0.4185	0.6177	0.8248	$+0.1391$
3	66	0.5692	0.5040	0.3269	0.6141	0.7104	$+0.0450$
4	51	0.3422	0.4314	0.4869	0.6431	0.7216	$+0.3009$
5	83	0.4279	0.4877	0.5032	0.6233	0.7768	$+0.1955$
6	66	0.4455	0.5606	0.6263	0.6540	0.8307	$+0.2086$

Stage dynamics. Table 12 gives the values behind Figure 4(a) for all subsets of the 400-question holdout, including all retrieval misses and the support-missing questions. Support-missing questions are those with at least one annotated supporting passage outside the top-5, whether or not an answer alias occurs in the retrieved context.

Table 12: Holdout F1 by retrieval label after each training stage. Bold marks the best stage.

Subset	n	Initial	Write	Recall	Protect	Centralized
All questions	400	0.4601	0.5052	0.4728	**0.6381**	0.7684
HotpotQA	200	0.5083	0.5129	0.4060	**0.6378**	–
2Wiki	200	0.4118	0.4974	0.5397	**0.6385**	–
Retrieval hits	237	0.5069	0.5474	0.4422	**0.7353**	–
Retrieval misses	163	0.3920	0.4438	**0.5174**	0.4968	0.6775
Support-missing questions	145	0.3220	0.3315	0.4293	**0.4844**	0.5764
Retrieval residuals	92	0.2598	0.2971	**0.4167**	0.3694	–

Alternative stage order. Training the six-client protocol in the order write, protect, recall, with the same client examples, initialization, aggregation, sample schedules, learning rates, and seed, reaches holdout F1 0.5052 after the write stage, 0.6232 after protection, and 0.5037 after the final recall stage, against 0.6381 for the fixed order, a paired difference of 0.1345 with a 95% CI of $[0.0929, 0.1761]$. On the main comparison the two orders differ by 0.0095 macro F1 with a 95% CI of $[-0.0114, 0.0308]$, the alternative lower on the retrieval hits, 0.5557 against 0.5990, and higher on the retrieval misses, 0.2364 against 0.2114.

Client count. Table 13 gives the values behind Figure 4(c), by benchmark and by retrieval label.

Table 13: Client-count scaling on the main comparison at a fixed total budget.

Clients	HotpotQA	2Wiki	PopQA	CWQ	Macro	Hits	Misses	Train (s)	Comm.
4	0.3027	0.4391	0.4451	0.4398	0.4067	0.5944	0.2145	1,458.22	1.85 GB
6	0.2989	0.4455	0.4288	0.4565	0.4074	0.5990	0.2114	1,460.77	2.78 GB
12	0.3117	0.4443	0.4312	0.4565	0.4109	0.6038	0.2135	1,490.23	5.56 GB

Same-question protocol by benchmark and answer presence. Table 14 gives the values behind Figure 2. A question is counted under answer in top-3 when a normalized alias of its reference answer occurs in the shared top-3 text, and Boolean questions are excluded from this split.

Table 14: Same-question comparison by benchmark, split by answer presence in the top-3. PRAG denotes Parametric RAG.

	All questions				Answer in top-3				Answer absent			
Benchmark	n	PRAG	No adapter	FedR^{2}M	n	PRAG	No adapter	FedR^{2}M	n	PRAG	No adapter	FedR^{2}M
HotpotQA	300	0.1966	0.2622	**0.2851**	107	0.3261	0.4601	**0.5430**	172	0.0762	0.0897	**0.1071**
2Wiki	300	0.2599	0.2254	**0.2658**	45	**0.5401**	0.2622	0.4000	225	0.1896	0.1903	**0.2300**
PopQA	300	0.1580	0.1722	**0.4296**	45	0.2364	0.2237	**0.5505**	255	0.1442	0.1631	**0.4083**
CWQ	300	0.4118	0.3810	**0.4671**	20	**0.7833**	0.7250	0.7750	280	0.3853	0.3564	**0.4451**
All	1,200	0.2566	0.2602	**0.3619**	217	0.3940	0.3945	**0.5363**	932	0.2150	0.2142	**0.3207**

Cost. Table 15 gives the strict cost measurements behind the analysis. Inference is measured on one A6000 GPU with fixed CPU affinity, 400 questions per setting, three repetitions, and a fixed 64-token answer, and the maximum coefficient of variation across repetitions is 0.0065 for inference and 0.0072 for training. The two inference settings are the plaintext top-5 path with and without the global adapter. The adapter changes no retrieval payload and adds model computation only.

Table 15: Strict cost measurements. Latencies are in milliseconds, the payload in bytes per question, inference peak memory in MiB, training wall clock in seconds, and training peak memory in GiB.

Inference setting	First token	64 tokens	Payload	Peak memory
Retrieval-only prompting, plaintext top-5	48.86	776.87	8,765.7	2,434.3
Plaintext top-5 with the global adapter	55.20	1,192.75	8,765.7	2,434.3
Training setting	**Wall clock**	**Updates**	**Peak memory**	**Communication**
Centralized	253.13	100	3.91	0
Six-client FedR^{2}M	1,473.01	612	3.91	2.78 GB

Offline cost of the document-adapter systems. Table 5 gives the offline cost of the document-adapter systems. The FedMosaic figures are the offline runtimes, artifact stores, and no-cache payload records of the frozen reproduction, one run per benchmark. The offline time covers clustering, adapter training, and mask training and excludes the synthetic question preparation, so it is a lower bound. The DistilledPRAG figure is the training log of the run behind Table 1. The FedR^{2}M figure covers training and excludes the string-matching discovery pass.

The 2Wiki exception. Table 16 splits the 300 2Wiki questions of the four-benchmark set into the 42 Boolean comparison questions and the rest, and Figure 5 shows the yes/no answers on the Boolean questions. The reference answers of the Boolean questions are 33 yes and 9 no. On all 300 questions retrieval residual memory remains 0.0871 above retrieval-only prompting with a 95% CI of $[0.0298, 0.1434]$. The Boolean comparison questions ask which of two entities satisfies an ordering, and memorized entity facts alone do not settle the ordering.

Table 16: 2Wiki in the four-benchmark set, split by answer type.

Subset	n	Retrieval only	No boundary	Permuted	Retrieval residual memory
All 2Wiki	300	0.4065	0.5328	0.4684	0.4936
Boolean comparison	42	0.2875	0.8095	0.2619	0.4048
Other	258	0.4259	0.4878	0.5020	0.5080

The four-benchmark set. Table 17 gives the per-benchmark view of the four-benchmark set discussed in the analysis.

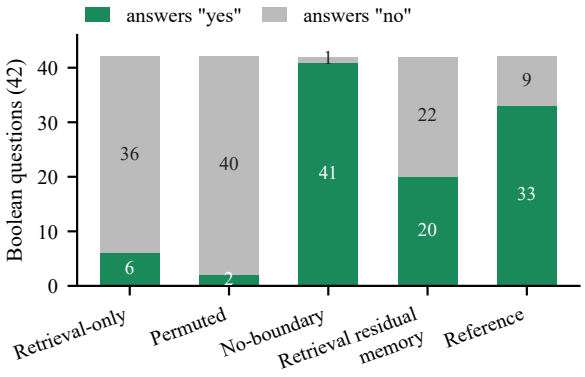

Figure 5: Yes/no answers on the 42 Boolean comparison questions of 2Wiki, against the reference.

Table 17: The four-benchmark set, with the four settings of the blind test.

Benchmark	Retrieval only	No boundary	Permuted	Retrieval residual memory
HotpotQA	0.2974	0.3894	0.4015	**0.4817**
2Wiki	0.4065	**0.5328**	0.4684	0.4936
PopQA	0.3057	0.3751	0.3622	**0.4781**
CWQ	0.3331	0.4836	0.4960	**0.5619**
Macro	0.3357	0.4452	0.4320	**0.5038**