# SYNFEDRAG: FEDERATED GRAPHRAG WITH PERSISTENT SYNTHETIC KNOWLEDGE GRAPH

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Federated learning (FL) enables LLM adaptation over decentralized private data, primarily by transferring knowledge through model parameters or adapters. However, for knowledge-intensive question answering (QA), relying solely on local information can be insufficient when relevant evidence is distributed across multiple clients. Retrieval-augmented generation (RAG) provides a natural mechanism for accessing such non-parametric knowledge rather than relying solely on knowledge encoded in model parameters. Existing RAG and GraphRAG systems typically assume centralized access to the underlying corpus or retrieval knowledge, making them difficult to apply when evidence is distributed across private clients. Recent federated RAG methods relax this by constructing shared retrieval memories from client-derived information. However, cross-client evidence may still need to be accessed remotely at query time, rather than being available as a persistent retrieval resource at each client. To address this gap, we propose SYNFEDRAG, which learns a shared synthetic graph that can be queried locally by every client without contacting other clients at inference time. The server constructs a shared synthetic knowledge graph from condensed graph optimized to approximate the retrieval behavior of clients' private graphs capturing distributed retrieval knowledge without centralizing raw documents or full local graphs. Clients then jointly read from their private and the shared synthetic knowledge graph, allowing them to access complementary retrieval knowledge beyond what is available in their local corpus. A lightweight graph-guided prompt module maps the retrieved subgraphs into soft prompts for a frozen LLM. During federated refinement, private QA supervision jointly adapts the graph-guided prompt module and continuously refines the shared synthetic knowledge graph over successive federated rounds. Experiments on non-IID decentralized multi-hop QA benchmarks show that SYNFEDRAG consistently improves answer quality and cross-client QA performance over local-only GraphRAG and federated LoRA baselines.

## 1 INTRODUCTION

Large language models (LLMs) have shown strong performance on knowledge-intensive tasks, but their parametric knowledge may be incomplete or outdated. Retrieval-augmented generation (RAG) grounds LLM responses in external evidence (Lewis et al., 2020), while GraphRAG further uses relations among documents, entities, and passages to support structured retrieval (Gutiérrez et al., 2024; He et al., 2024; Zhuang et al., 2025; Hu et al., 2025). However, most existing RAG and GraphRAG systems assume that the underlying retrieval corpus or graph memory is centrally accessible (Lewis et al., 2020; Gutiérrez et al., 2024; Zhuang et al., 2025; Hu et al., 2025). This assumption is restrictive in practical settings, where relevant documents are distributed across organizations or users and cannot be directly pooled due to ownership, confidentiality, regulatory, or data-governance constraints. Without a mechanism for sharing retrieval knowledge, each client can retrieve only from its private corpus. Consequently, local retrieval may provide insufficient evidence for questions whose supporting information is distributed across clients.

Federated learning (FL) provides a natural mechanism for decentralized collaboration, but exchanging model updates does not by itself provide a shared retrieval memory. Conventional federated LLM methods (see Appendix A for more details) transfer knowledge primarily through model updates or parameter-efficient modules such as prompts or adapters, whereas the evidence for RAG re-



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.

mains stored in client-side non-parametric retrieval structures (Che et al., 2023; Wang et al., 2024b; Cho et al., 2024; Singhal et al., 2025; Bian et al., 2025). Existing federated RAG methods address this gap through distributed search, personalized retrieval, prototype alignment (Wang et al., 2024a; Guerraoui et al., 2025; He et al., 2025; Yang et al., 2026; Zhou et al., 2025; Gao et al., 2026a). Yet existing structured approaches either require query-time access to remote repositories or do not learn a persistent shared graph whose representations and topology are jointly optimized for local cross-client retrieval. Consequently, a key question remains: *How can we construct a persistent shared graph retrieval resource that transfers knowledge across clients while remaining locally queryable without contacting other clients at inference time?*

Motivated by this gap, we propose **SynFedRAG**, a federated GraphRAG framework that learns a shared synthetic graph for retrieval-augmented question answering over decentralized private corpora. Each client first constructs a local entity-sentence-passage tri-partite graph and condenses it into a compact summary. The server aggregates these summaries to initialize a shared synthetic knowledge graph that captures information distributed across clients. During federated refinement, clients retrieve from their private graph and read from the shared synthetic graph, Private QA supervision updates the graph-guided prompt module and refines the shared synthetic graph across communication rounds. The LLM remains frozen throughout, while raw documents and complete local graphs remain on the clients.

The key idea behind SynFedRAG is to decouple *knowledge sharing* from *raw corpus sharing*. Rather than reproducing or centralizing complete client graphs, SynFedRAG combines condensed local summaries into a shared synthetic graph that integrates complementary knowledge across clients. Each client can query this graph together with its private graph, providing access to cross-client context while the raw corpora remain local. Consequently, this enables SynFedRAG to support persistent cross-client retrieval without contacting other clients at inference time. The main contributions are summarized as

- We introduce the setting of federated GraphRAG with a persistent, locally queryable shared retrieval structure, as opposed to query-time remote retrieval, and identify the shared synthetic graph as the object to be learned.
- To avoid transmitting complete local retrieval graphs, we use client-side graph condensation to produce compact summaries for synthetic-graph initialization, reducing one-time graph communication to $4.88\%$–$35.49\%$ of full-graph transmission across the evaluated budgets.
- We introduce a shared synthetic graph constructed from condensed summaries contributed by multiple clients and refined through federated QA supervision. Each client can query this graph together with its private graph without accessing other clients at inference time.
- We design a federated graph-guided prompt tuning strategy that jointly adapts a lightweight GNN encoder and projection module to transform retrieved local and synthetic graph evidence into soft prompts for a frozen LLM.
- We evaluate SynFedRAG under random and topic-skewed federated corpus partitions, examining overall QA, cross-client QA, condensation fidelity, component contributions, and communication cost.

## 2 METHOD

### 2.1 PROBLEM SETUP AND FRAMEWORK OVERVIEW

We consider a federated question-answering system consisting of one server and $M$ clients. Client $m \in \{1, \ldots, M\}$ owns a private document corpus $\mathcal{D}_m$ and a QA dataset $\mathcal{Q}_m = \{(q_i^{(m)}, a_i^{(m)})\}_{i=1}^{n_m}$, where $q_i^{(m)}$ and $a_i^{(m)}$ denote the $i$-th question and its corresponding answer, respectively. Following LinearRAG Zhuang et al. (2025), each client maintains a private tri-partite retrieval graph $G_m = (V_m, E_m, X_m)$, where $V_m = V_m^E \cup V_m^S \cup V_m^P$ denotes the entity, sentence, and passage nodes, respectively. The edge set $E_m = E_m^{ES} \cup E_m^{EP}$ contains entity-sentence and entity-passage relations. The node-feature matrix $\mathbf{X}_m$ contains text-encoder representations of the corresponding entity, sentence, and passage contents. The private corpus $\mathcal{D}_m$ and complete graph $G_m$ remain local

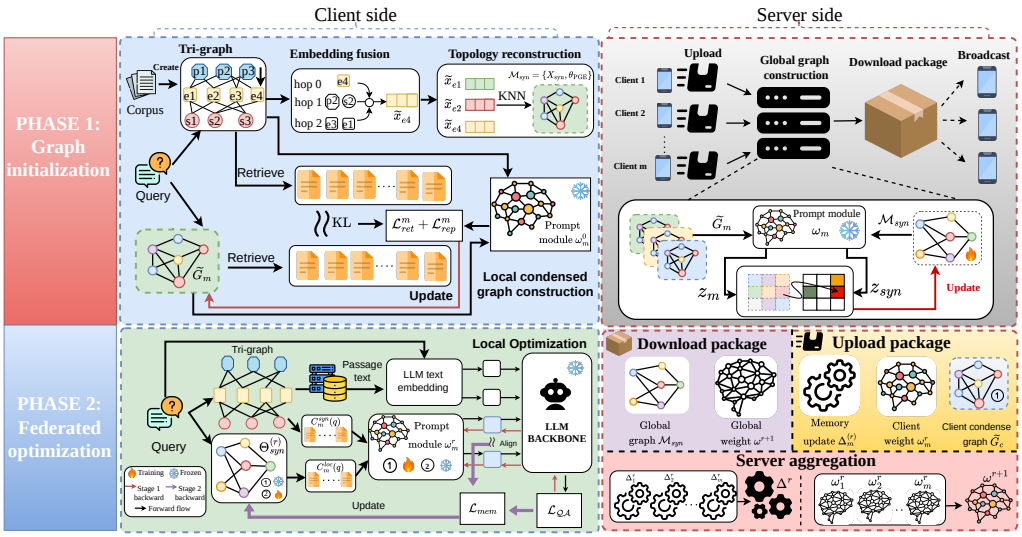



Figure 1: Overview of SynFedRAG. During initialization, clients condense private tri-graphs into compact summaries used to initialize a shared synthetic graph. During federated refinement, clients retrieve local evidence, update the graph-guided prompt module and synthetic graph, and the server aggregates and regularizes these updates before broadcasting the global model.

throughout training. This locality prevents a client from directly retrieving complementary evidence stored in other clients' corpora. Table 7 summarizes the main notation.

To circumvent this restriction, SynFedRAG operates via the two-phase framework illustrated in Fig. 1. *Phase I* constructs an initial shared synthetic knowledge graph without centralizing complete private graphs. Each client condenses its private graph into a compact summary optimized to retain its retrieval behavior, and the server aggregates these summaries to initialize the shared synthetic graph. However, preserving graph information alone does not ensure that the resulting graph is useful for downstream question answering. *Phase II* therefore refines the shared graph and the graph-guided prompt module via federated QA supervision, allowing clients to exploit complementary local and cross-client knowledge. Throughout both phases, raw documents, complete private graphs, QA pairs, and query-specific retrieval traces remain local, although the uploaded condensed graphs can reveal passage membership to the server Appendix D.4.

### 2.2 Phase I: Synthetic Graph Initialization

Phase I transforms decentralized private corpora into a compact shared retrieval resource without centralizing the underlying documents or complete client graphs. It consists of client-side graph construction and condensation, followed by server-side synthetic knowledge graph initialization.

#### 2.2.1 Client-Side Private Graph Construction and Condensation

Client $m$ condenses its private graph $G_m$ into a compact graph $\widetilde{G}_m = (\widetilde{V}_m, \widetilde{E}_m, \widetilde{X}_m)$ with $|\widetilde{V}_m| \ll |V_m|$. This condensation aims to reduce communication while retaining information needed for retrieval. We select a fixed budget of entity anchors using an importance-aware maximal marginal relevance criterion and expand each anchor with neighboring sentence and passage nodes (see Appendix B.2.1 for more details). Their neighborhood information is used to initialize the condensed node features, while inherited and semantic-neighbor relations construct the condensed topology. After construction, $\widetilde{V}_m$ and $\widetilde{E}_m$ remain fixed, and only the condensed features are refined:

$$\widetilde{X}_m^\star = \arg\min_{\widetilde{X}_m} \mathcal{L}_{\text{cond}}^{(m)}, \qquad \mathcal{L}_{\text{cond}}^{(m)} = \mathcal{L}_{\text{ret}}^{(m)} + \lambda_{\text{rep}}\mathcal{L}_{\text{rep}}^{(m)} + \lambda_{\text{div}}\mathcal{L}_{\text{div}}^{(m)}. \tag{1}$$

Here, $\mathcal{L}_{\text{ret}}^{(m)}$ preserves the retrieval behavior of the complete private graph, $\mathcal{L}_{\text{rep}}^{(m)}$ preserves its semantic and contextual representations, and $\mathcal{L}_{\text{div}}^{(m)}$ discourages redundant condensed nodes. Detailed construction and objective formulations are provided in Appendix B.2. The resulting $\widetilde{G}_m$ is uploaded once to the server for synthetic graph initialization.

### 2.2.2 SERVER-SIDE SYNTHETIC GRAPH INITIALIZATION

Given the uploaded client-condensed graphs $\{\widetilde{G}_m\}_{m=1}^{M}$, the server first constructs a fixed budget of $K_g$ synthetic nodes with learnable representations $\mathbf{X}_{\text{syn}} \in \mathbb{R}^{K_g \times d_x}$, where $d_x$ is the node-feature dimension shared with the client graphs. Unlike nodes in the client-condensed graphs, these synthetic nodes do not inherit a predefined topology because they are not in one-to-one correspondence with nodes from any individual client. A connectivity structure must therefore be inferred for graph-based processing.

Treating the synthetic adjacency and node representations as independent free variables can overlook their implicit dependence and introduces a quadratic number of structural parameters (Jin et al., 2022). Following prior graph-condensation approaches (Jin et al., 2022), we leverage a parameterized graph estimator (PGE), parameterized by $\boldsymbol{\theta}_{\text{PGE}}$, to infer the synthetic adjacency from the current node representations $\mathbf{A}_{\text{syn}} = \text{PGE}_{\boldsymbol{\theta}_{\text{PGE}}}(\mathbf{X}_{\text{syn}})$. This parameterization couples the synthetic topology to its node representations, allowing the connectivity to adapt as $\mathbf{X}_{\text{syn}}$ is refined. The resulting synthetic graph is represented as $\mathcal{M}_{\text{syn}}^{(0)} = \{\mathbf{X}_{\text{syn}}, \boldsymbol{\theta}_{\text{PGE}}\}$.

Graph operations on $\mathcal{M}_{\text{syn}}$ use this induced topology. Detailed PGE construction and synthetic-graph regularization are provided in Appendix B.3. To initialize the synthetic graph from heterogeneous client summaries, we align their representations with the synthetic graph in LLM embedding space. We compute the initial alignment using a randomly initialized graph-guided prompt module $W^{(0)} = \{\theta_{\text{GNN}}^{(0)}, \boldsymbol{\theta}_{\text{proj}}^{(0)}\}$, comprising a GNN encoder and a projection network, as follows:

$$z_m = g_{\text{proj}}\left(\text{Pool}\left[f_{\text{GNN}}(\widetilde{G}_m; \boldsymbol{\theta}_{\text{GNN}}^{(0)})\right]; \boldsymbol{\theta}_{\text{proj}}^{(0)}\right), \quad z_{\text{syn}} = g_{\text{proj}}\left(\text{Pool}\left[f_{\text{GNN}}(\mathcal{M}_{\text{syn}}; \boldsymbol{\theta}_{\text{GNN}}^{(0)})\right]; \boldsymbol{\theta}_{\text{proj}}^{(0)}\right). \tag{2}$$

Here, $z_m, z_{\text{syn}} \in \mathbb{R}^{d_{\text{LLM}}}$ denote the graph-level embeddings of $\widetilde{G}_m$ and $M_{\text{syn}}$, respectively. Specifically, $f_{\text{GNN}}$ encodes node features and graph connectivity into node representations, which are mean-pooled across nodes to obtain a graph-level representation, while $g_{\text{proj}} : \mathbb{R}^{d_g} \to \mathbb{R}^{d_{\text{LLM}}}$ maps the resulting graph representation into the embedding space expected by the frozen LLM. Following G-Retriever (He et al., 2024), representations in this space can subsequently be inserted as soft graph token alongside LLM's textual token embeddings. Because the client-condensed and synthetic graphs differ in size and topology and have no explicit node-level correspondence, we align their graph-level representations by minimizing

$$\mathcal{L}_{\text{align}}^{\text{init}} = \frac{1}{M}\sum_{m=1}^{M}\left[1 - \cos\left(z_m, z_{\text{syn}}\right)\right]. \tag{3}$$

Because $\mathcal{L}_{\text{align}}^{\text{init}}$ constrains only the pooled graph representation, different synthetic node features and connectivity patterns can produce the same graph-level representation. This underdetermination can produce redundant synthetic nodes, reducing the effective capacity of the synthetic graph. We therefore penalize excessive pairwise cosine similarity among the synthetic-node features:

$$\mathcal{L}_{\text{reg}} = \frac{\lambda_{\text{reg}}}{K_g(K_g-1)}\sum_{i\neq j}\left[\cos\left(\mathbf{x}_i^{\text{syn}}, \mathbf{x}_j^{\text{syn}}\right) - \delta_{\text{syn}}\right]_+. \tag{4}$$

Here, $[u]_+ = \max(0, u)$, $\delta_{\text{syn}}$ is the similarity threshold, and $\lambda_{\text{reg}}$ controls the regularization strength. The initialized synthetic graph is then obtained as

$$\mathcal{M}_{\text{syn}}^{(0)} = \arg\min_{\mathcal{M}_{\text{syn}}}\left(\mathcal{L}_{\text{align}}^{\text{init}} + \mathcal{L}_{\text{reg}}\right), \tag{5}$$

During initialization, the uploaded client-condensed graphs remain unchanged. We also freeze the encoder and projector $W^{(0)}$, so their encoded representations $z_m$ remain unchanged while we optimize the synthetic graph to match these targets.

## 2.3 PHASE II: FEDERATED SYNTHETIC GRAPH REFINEMENT

Phase I provides a shared synthetic-graph initialization from client-condensed representations, but neither the synthetic graph nor the graph-guided prompt module is directly optimized for downstream question answering. Moreover, the global initialization may smooth over client-specific information. Phase II therefore locally refines both the synthetic graph and the graph-guided prompt

module using private QA and retrieval signals, followed by server aggregation of the resulting updates. At communication round $r$, the server broadcasts the global graph-guided prompt module $W^{(r)}$ and synthetic graph $\mathcal{M}_{\text{syn}}^{(r)}$ to a set of participating clients $\mathcal{S}_r$. Each participating client performs local QA adaptation and synthetic graph refinement, and returns only the resulting prompt-module and graph updates.

#### 2.3.1 Client-Side Retrieval and Local Adaptation

For each query $q$, client $m$ retrieves a local evidence subgraph and accesses the shared synthetic graph through a query-conditioned soft readout. We denote the corresponding graph contexts by:

$$\mathcal{C}_m^{\text{loc}}(q) = \text{Retrieve}(q, G_m), \qquad \mathcal{C}_m^{\text{syn}}(q) = \text{Retrieve}\left(q, \mathcal{M}_{\text{syn}}^{(r)}\right). \tag{6}$$

Here, $\mathcal{C}_m^{\text{loc}}(q)$ captures evidence available in client $m$'s private corpus, whereas $\mathcal{C}_m^{\text{syn}}(q)$ provides complementary knowledge transferred through the shared synthetic graph. Details of local and synthetic retrieval are provided in Appendix B.4.The local and synthetic branches share the GNN encoder and projector in $W^{(r)}$, with different pooling operations. We then have

$$z_q^s = g_{\text{proj}}\left(\text{Pool}\left[f_{\text{GNN}}\left(\mathcal{C}_m^s(q); \boldsymbol{\theta}_{\text{GNN}}^{(r)}\right)\right]; \boldsymbol{\theta}_{\text{proj}}^{(r)}\right), \quad \text{where } s \in \{\text{loc}, \text{syn}\}. \tag{7}$$

The GNN encodes node features and connectivity into node representations. We mean-pool these representations over the retrieved local subgraph. For the synthetic graph, we aggregate them using question-conditioned weights computed from the cosine similarity between the question embedding and synthetic-node features defined in Appendix B.4.2. The shared projector maps each pooled representation into a soft graph token, and both tokens are incorporated into the frozen LLM input:

$$\mathbf{I}_m(q, a) = [\text{BOS}, z_q^{\text{loc}}, z_q^{\text{syn}}, \mathbf{T}_{\text{desc}}^{[:L]}, \mathbf{T}_q, \text{EOS}_{\text{user}}, \mathbf{T}_a, \text{EOS}], \tag{8}$$

where $\mathbf{T}_q$ and $\mathbf{T}_a$ denote the question and answer token embeddings, and $\mathbf{T}_{\text{desc}}^{[:L]}$ denotes retrieved local passage text truncated to length $L$. The corresponding QA objective is

$$\mathcal{L}_{\text{QA}}^{(m)} = -\sum_{(q,a)\in\mathcal{Q}_m} \sum_{t=1}^{|a|} \log p_{\boldsymbol{\theta}_{\text{LLM}}}\left(a_t \mid \mathbf{I}_m(q, a)_{<t}\right). \tag{9}$$

The LLM parameters $\boldsymbol{\theta}_{\text{LLM}}$ remain frozen. Starting from the broadcast prompt module $W^{(r)}$, client $m$ locally minimizes Eq. (9) to obtain the adapted parameters $W_m^{(r)}$.

**Local synthetic graph refinement.** Phase-I alignment provides a shared initialization, but may average out client-specific information. Phase II therefore refines a local copy at each client using representation alignment and gradient matching, which respectively align the synthetic graph with the client's condensed graph and its QA-relevant behavior with private evidence. In addition to adapting the graph-guided prompt module, client $m$ refines its local synthetic-graph copy using

$$\mathcal{L}_{\text{mem}}^{(m)} = \mathcal{L}_{\text{QA}}^{(m)} + \lambda_{\text{gm}}\mathcal{L}_{\text{GM}}^{(m)} + \lambda_{\text{align}}\mathcal{L}_{\text{align}}^{(m)}. \tag{10}$$

Here, $\mathcal{L}_{\text{align}}^{(m)}$ preserves consistency between the local synthetic graph copy and client $m$'s condensed graph, while $\mathcal{L}_{\text{GM}}^{(m)}$ encourages the synthetic context to induce QA behavior consistent with that induced by private local evidence. The detailed formulations are provided in Appendix B.4. The $m$-th client itself then updates the synthetic knowledge graph as

$$\mathcal{M}_{\text{syn},m}^{(r)} = \mathcal{M}_{\text{syn},m}^{(r)} - \eta_{\text{syn}}\nabla_{\mathcal{M}_{\text{syn}}}\mathcal{L}_{\text{mem}}^{(m)}. \tag{11}$$

After local refinement, client $m$ forms the graph update $\Delta_m^{(r)} = \mathcal{M}_{\text{syn},m}^{(r)} - \mathcal{M}_{\text{syn}}^{(r)}$, where the difference is taken over the trainable components of $\mathcal{M}_{\text{syn}}$. Since all participating clients start from the same broadcast synthetic graph, $\Delta_m^{(r)}$ isolates the client-specific refinement induced by its private data and can be directly aggregated at the server. The client uploads $W_m^{(r)}$ and $\Delta_m^{(r)}$ to the server, while its raw documents, QA pairs, private graph, and query-specific retrieval contexts remain local.

#### 2.3.2 SERVER-SIDE AGGREGATION AND GLOBAL UPDATE

After local adaptation, the server receives $\{W_m^{(r)}, \Delta_m^{(r)}\}_{m\in\mathcal{S}_r}$ from the participating clients $\mathcal{S}_r$. The graph-guided prompt module and the client synthetic graph updates are aggregated using weighted averaging:

$$W^{(r+1)} = \sum_{m\in\mathcal{S}_r} \frac{n_m}{\sum_{j\in\mathcal{S}_r} n_j} W_m^{(r)}, \quad \Delta^{(r)} = \sum_{m\in\mathcal{S}_r} \frac{n_m}{\sum_{j\in\mathcal{S}_r} n_j} \Delta_m^{(r)}. \tag{12}$$

The server then updates the shared synthetic graph according to

$$\mathcal{M}_{\text{syn}}^{(r+1)} = \mathcal{M}_{\text{syn}}^{(r)} + \Delta^{(r)} - \eta_{\text{reg}} \nabla_{\mathcal{M}_{\text{syn}}} \mathcal{L}_{\text{reg}}, \tag{13}$$

where $\mathcal{L}_{\text{reg}}$ is the synthetic-node feature regularizer defined in Eq. (4). This step discourages excessive similarity among synthetic-node features across communication rounds. The gradient with respect to $\mathcal{M}_{\text{syn}}$ is understood to act on its trainable components $\mathbf{X}_{\text{syn}}$ and $\boldsymbol{\theta}_{\text{PGE}}$.

Table 1: Performance comparison against baselines under random partition. The best, second-best, and third-best results are highlighted in green, blue, and yellow, respectively.

| Group | Method | HotpotQA | | MuSiQue | | 2WikiMultiHopQA | |
|---|---|---|---|---|---|---|---|
| | | EM | F1 | EM | F1 | EM | F1 |
| Federated LLM | FLoRA | $15.94 \pm 0.42$ | $23.71 \pm 0.38$ | $1.90 \pm 0.18$ | $10.69 \pm 0.31$ | $19.20 \pm 0.35$ | $21.87 \pm 0.41$ |
| | FlexLoRA | $22.31 \pm 0.37$ | $30.30 \pm 0.45$ | $3.12 \pm 0.21$ | $10.79 \pm 0.34$ | $21.11 \pm 0.39$ | $25.44 \pm 0.43$ |
| | RoLoRA | $16.23 \pm 0.41$ | $24.84 \pm 0.36$ | $5.84 \pm 0.27$ | $11.63 \pm 0.39$ | $20.34 \pm 0.44$ | $23.42 \pm 0.40$ |
| FL + Local GraphRAG | HippoRAG | $28.01 \pm 0.53$ | $37.16 \pm 0.61$ | $7.01 \pm 0.32$ | $16.75 \pm 0.47$ | $29.29 \pm 0.48$ | $32.57 \pm 0.52$ |
| | LinearRAG | $27.52 \pm 0.49$ | $34.77 \pm 0.55$ | $6.51 \pm 0.29$ | $15.06 \pm 0.43$ | $25.26 \pm 0.51$ | $29.18 \pm 0.49$ |
| | G-retriever | $31.87 \pm 0.57$ | $36.77 \pm 0.58$ | $8.67 \pm 0.35$ | $18.59 \pm 0.49$ | $27.57 \pm 0.46$ | $31.98 \pm 0.54$ |
| | GRAG | $28.33 \pm 0.50$ | $37.75 \pm 0.63$ | $8.63 \pm 0.34$ | $18.26 \pm 0.45$ | $25.77 \pm 0.47$ | $30.55 \pm 0.51$ |
| FL + Local RAG | FLARE | $28.97 \pm 0.52$ | $39.30 \pm 0.59$ | $8.77 \pm 0.37$ | $17.88 \pm 0.48$ | $32.20 \pm 0.55$ | $36.84 \pm 0.57$ |
| | Dense RAG | $27.14 \pm 0.34$ | $39.87 \pm 0.57$ | $9.87 \pm 1.34$ | $16.67 \pm 1.15$ | $29.12 \pm 0.98$ | $32.52 \pm 0.82$ |
| Federated RAG | DGRAG | $32.60 \pm 0.11$ | $42.17 \pm 0.26$ | $26.03 \pm 0.34$ | $31.38 \pm 0.17$ | $33.20 \pm 0.23$ | $37.68 \pm 0.19$ |
| | FD-RAG | $33.84 \pm 0.14$ | $42.30 \pm 0.67$ | $27.15 \pm 0.26$ | $30.04 \pm 0.32$ | $30.07 \pm 0.14$ | $37.04 \pm 0.13$ |
| | **SynFedRAG (Ours)** | $\mathbf{37.34 \pm 0.15}$ | $\mathbf{47.43 \pm 0.13}$ | $\mathbf{32.02 \pm 1.03}$ | $\mathbf{35.61 \pm 0.21}$ | $\mathbf{36.00 \pm 0.24}$ | $\mathbf{40.85 \pm 0.47}$ |

Finally, the updated package $\{W^{(r+1)}, \mathcal{M}_{\text{syn}}^{(r+1)}\}$ is broadcast to the participating clients in the next communication round. Iterating the client and server updates progressively adapts both the graph-to-LLM interface and the shared synthetic graph to decentralized QA supervision.

Table 3: Fidelity of client-side graph condensation under different node budgets on HotpotQA, random partition. Relative communication denotes the one-time cost relative to transmitting the complete private graph.

| Node Ratio | Edge Ratio | Retrieval Agreement | Passage Recall R@5 | Passage Recall R@10 | QA F1 | Relative Comm. |
|---|---|---|---|---|---|---|
| 0.2 | 48.26 | 85.60 | **62.10** | **67.35** | **47.43** | 35.49 |
| 0.1 | 25.06 | 69.56 | 49.45 | 53.75 | 46.83 | 22.48 |
| 0.05 | 12.60 | 45.92 | 31.40 | 33.85 | 43.56 | 12.63 |
| 0.02 | 5.02 | 18.34 | 12.85 | 13.55 | 40.98 | 4.88 |

## 3 EXPERIMENTAL SETTINGS

We summarize the main experimental settings here and provide full implementation and evaluation details in Appendix C.

**Datasets.** We evaluate on three multi-hop QA benchmarks: HotpotQA Yang et al. (2018), MuSiQue Trivedi et al. (2022), and 2WikiMultiHopQA Ho et al. (2020). Following the benchmark evaluation setup of HippoRAG Gutiérrez et al. (2024), we use 1,000 questions from each dataset's validation split as the test set. A separate pool of 1,000 questions is divided into 800 training and 200 validation questions. For each dataset, we construct a merged retrieval corpus from the candidate passages associated with both question sets, including supporting and distractor passages, before partitioning it among three clients. The resulting corpora contain 19,583 passages for HotpotQA, 25,883 for MuSiQue, and 11,335 for 2WikiMultiHopQA. Although test-associated passages are included in the corpus, test questions and answers are excluded from training.

Table 2: F1 performance on multi-hop QA benchmarks under topic-skewed federated partitions. Smaller $\alpha$ corresponds to stronger topic heterogeneity. The best, second-best, and third-best reported means are highlighted in green, blue, and yellow, respectively.

| | | HotpotQA | | MuSiQue | | 2WikiMultiHopQA | |
|---|---|---|---|---|---|---|---|
| **Group** | **Method** | $\alpha = 1.0$ | $\alpha = 0.1$ | $\alpha = 1.0$ | $\alpha = 0.1$ | $\alpha = 1.0$ | $\alpha = 0.1$ |
| Federated LLM | FLoRA | $20.17 \pm 0.84$ | $22.03 \pm 1.16$ | $8.35 \pm 0.47$ | $7.11 \pm 0.62$ | $30.17 \pm 1.23$ | $31.24 \pm 1.08$ |
| | FlexLoRA | $21.26 \pm 1.07$ | $21.73 \pm 0.93$ | $7.37 \pm 0.58$ | $6.89 \pm 0.43$ | $33.65 \pm 0.96$ | $29.21 \pm 1.42$ |
| | RoLoRA | $22.86 \pm 0.91$ | $22.32 \pm 1.28$ | $8.48 \pm 0.39$ | $8.20 \pm 0.71$ | $29.55 \pm 1.34$ | $35.16 \pm 1.17$ |
| FL + Local GraphRAG | HippoRAG | $35.27 \pm 1.12$ | $38.40 \pm 0.97$ | $19.73 \pm 0.76$ | $18.22 \pm 1.04$ | $35.98 \pm 0.88$ | $36.24 \pm 1.31$ |
| | LinearRAG | $36.94 \pm 0.86$ | $37.18 \pm 1.27$ | $15.37 \pm 0.64$ | $15.82 \pm 0.53$ | $30.11 \pm 1.09$ | $30.74 \pm 0.92$ |
| | G-retriever | $38.24 \pm 1.25$ | $34.51 \pm 1.48$ | $19.37 \pm 0.83$ | $19.11 \pm 0.69$ | $34.92 \pm 1.16$ | $34.84 \pm 1.37$ |
| | GRAG | $37.56 \pm 0.94$ | $32.43 \pm 1.36$ | $20.11 \pm 0.72$ | $18.42 \pm 0.98$ | $36.73 \pm 1.27$ | $33.63 \pm 1.05$ |
| FL + Local RAG | FLARE | $29.56 \pm 1.18$ | $27.72 \pm 1.43$ | $12.67 \pm 0.56$ | $14.92 \pm 0.81$ | $30.28 \pm 0.93$ | $31.93 \pm 1.21$ |
| | Dense RAG | $36.14 \pm 1.04$ | $26.37 \pm 1.57$ | $10.52 \pm 0.68$ | $11.71 \pm 0.59$ | $31.13 \pm 1.14$ | $32.34 \pm 0.99$ |
| Federated RAG | DGRAG | $44.17 \pm 0.92$ | $47.64 \pm 1.24$ | $31.96 \pm 0.87$ | $35.01 \pm 0.74$ | $38.05 \pm 1.06$ | $42.69 \pm 1.33$ |
| | FD-RAG | $42.39 \pm 1.15$ | $45.17 \pm 0.98$ | $33.14 \pm 0.65$ | $34.84 \pm 0.93$ | $38.21 \pm 0.89$ | $41.54 \pm 1.19$ |
| | **SynFedRAG (Ours)** | $\mathbf{47.71} \pm 1.03$ | $\mathbf{52.10} \pm 1.29$ | $\mathbf{34.78} \pm 0.79$ | $\mathbf{35.14} \pm 0.88$ | $\mathbf{41.37} \pm 1.12$ | $\mathbf{45.70} \pm 0.84$ |

# 4 Experimental Evaluations

## 4.1 Overall Question Answering Performance

Table 1 compares SynFedRAG with federated LLM adaptation, local retrieval, and federated RAG baselines. Across all three datasets, retrieval-augmented methods consistently outperform parameter-only federated adaptation, highlighting the importance of explicit access to external evidence for knowledge-intensive QA. Federated RAG methods further improve over local RAG and GraphRAG baselines, with the largest gains observed on MuSiQue, whose connected 2–4-hop reasoning may make missing intermediate evidence particularly consequential Trivedi et al. (2022). SynFedRAG achieves the best performance on all benchmarks, reaching F1 scores of $47.43$, $35.61$, and $40.85$ on HotpotQA, MuSiQue, and 2WikiMultiHopQA, respectively. Compared with the strongest competing federated RAG baseline, these results show that SynFedRAG more effectively supports QA over decentralized retrieval settings across all three benchmarks.

Table 2 reports performance under topic-skewed corpus partitions. Several methods achieve higher F1 at $\alpha = 0.1$, but this comparison reflects changes in both topic heterogeneity and evidence availability. Although the test questions and their assigned clients remain fixed, repartitioning the corpus changes which supporting passages are available locally. As shown in Table 8, the local-answerable share increases from 14.6% to 18.0% on HotpotQA and from 10.4% to 14.8% on 2WikiMultiHopQA. This increased local support coverage may partly explain the higher overall scores; these results therefore do not establish that stronger non-IID heterogeneity itself improves performance.

## 4.2 Graph Condensation Fidelity

Table 3 shows that stronger graph compression reduces retrieval fidelity, with retrieval agreement dropping from $85.60$ to $18.34$ and R@10 from $67.35$ to $13.55$ as the retained node ratio decreases from $20\%$ to $2\%$. Despite this, QA F1 decreases more modestly, from $47.43$ to $40.98$. These results indicate that, across the evaluated compression budgets, final QA performance is less sensitive to compression than the measured retrieval fidelity. Meanwhile, the one-time graph communication decreases from $35.49\%$ to $4.88\%$ of full-graph transmission.

## 4.3 Component and Objective Ablations

Table 4 evaluates the contributions of synthetic memory, initialization, and refinement objectives under topic-skewed partitions with $\alpha = 1.0$. The full model achieves the highest F1 across all three benchmarks. Using only local retrieval or only synthetic memory yields lower performance than combining both sources. Removing Phase-I initialization or Phase-II refinement also reduces F1, supporting the use of both stages. Each individual objective ablation lowers the reported mean F1, although the magnitude varies across datasets.

### 4.4 CROSS-CLIENT KNOWLEDGE TRANSFER

We evaluate cross-client QA under topic-skewed partitions with $\alpha = 1.0$ by separating test questions according to the availability of their annotated supporting passages at the assigned client. Table 5 reports F1 separately for local-answerable and cross-client questions. Most local retrieval baselines degrade on the cross-client subset. Here, SynFedRAG achieves cross-client F1 scores of 44.70, 34.53, and 41.08 on HotpotQA, MuSiQue, and 2WikiMultiHopQA, respectively, exceeding the strongest competing federated RAG baseline by 1.48, 1.36, and 3.32 points. These results show that SynFedRAG remains effective when supporting evidence is distributed across clients. This advantage is consistent with the role of the shared synthetic knowledge graph. Specifically, removing the synthetic graph, F1 then decreases by 13.87, 20.15, and 11.18 points across the three benchmarks. We additionally evaluate a topic-matched question-ownership ablation, which better aligns client queries with local corpus specialization; the corresponding support statistics and local/cross-client F1 are reported in Appendix D.3.

Table 4: Ablation of synthetic-graph construction and refinement. Overall F1 under topic-skewed partitions with $\alpha = 1.0$. Higher is better.

| Variant | HotpotQA | MuSiQue | 2Wiki |
|---|---|---|---|
| *Shared-graph controls* | | | |
| w/o synthetic graph | $33.84 \pm 1.04$ | $14.63 \pm 0.11$ | $30.19 \pm 0.21$ |
| Synthetic graph only | $37.18 \pm 0.92$ | $11.69 \pm 0.37$ | $20.62 \pm 0.74$ |
| *Synthetic-topology controls* | | | |
| Self-loops only ($A_{\text{syn}} = I$) | $35.24 \pm 0.84$ | $18.62 \pm 0.49$ | $32.40 \pm 0.62$ |
| Fixed kNN topology | $41.57 \pm 0.72$ | $25.68 \pm 1.19$ | $37.91 \pm 0.43$ |
| No topology | $34.94 \pm 1.37$ | $16.23 \pm 0.54$ | $32.61 \pm 1.03$ |
| *Phase I: Synthetic-graph initialization* | | | |
| w/o Phase-I initialization | $34.20 \pm 0.56$ | $11.30 \pm 0.78$ | $28.87 \pm 0.92$ |
| *Phase II: Federated graph refinement* | | | |
| w/o Phase-II refinement | $40.27 \pm 1.74$ | $21.30 \pm 0.12$ | $32.19 \pm 0.63$ |
| w/o $\mathcal{L}_{\text{QA}}$ for graph | $41.13 \pm 0.61$ | $22.34 \pm 0.38$ | $32.84 \pm 0.57$ |
| w/o $\mathcal{L}_{\text{GM}}$ | $43.19 \pm 1.24$ | $24.97 \pm 0.89$ | $32.11 \pm 0.36$ |
| w/o local $\mathcal{L}_{\text{align}}$ | $42.17 \pm 0.56$ | $25.06 \pm 0.53$ | $36.94 \pm 0.21$ |
| w/o $\mathcal{L}_{\text{reg}}$ | $44.19 \pm 0.97$ | $32.26 \pm 1.12$ | $37.12 \pm 0.84$ |
| **SynFedRAG** | $\mathbf{47.71 \pm 1.03}$ | $\mathbf{34.78 \pm 0.79}$ | $\mathbf{41.37 \pm 1.12}$ |

Table 5: F1 comparison on local-answerable and cross-client questions under topic-skewed partitions with $\alpha = 1.0$. Higher is better. The best, second-best, and third-best means are highlighted in green, blue, and yellow, respectively.

| Group | Method | HotpotQA | | MuSiQue | | 2Wiki | |
|---|---|---|---|---|---|---|---|
| | | Local | Cross | Local | Cross | Local | Cross |
| Federated LLM | FLoRA | $24.19 \pm 1.12$ | $19.46 \pm 0.84$ | $11.17 \pm 0.83$ | $8.06 \pm 0.46$ | $21.98 \pm 0.91$ | $31.15 \pm 1.27$ |
| | FlexLoRA | $30.78 \pm 1.37$ | $19.58 \pm 1.04$ | $11.22 \pm 0.69$ | $6.98 \pm 0.65$ | $25.94 \pm 1.08$ | $34.57 \pm 1.18$ |
| | RoLoRA | $26.79 \pm 0.96$ | $22.17 \pm 0.94$ | $16.37 \pm 1.14$ | $7.67 \pm 0.72$ | $24.20 \pm 0.87$ | $30.19 \pm 0.89$ |
| FL + Local GraphRAG | HippoRAG | $33.20 \pm 1.46$ | $35.64 \pm 1.37$ | $20.32 \pm 1.07$ | $19.67 \pm 0.84$ | $43.52 \pm 1.23$ | $35.08 \pm 0.89$ |
| | LinearRAG | $46.20 \pm 1.28$ | $35.31 \pm 0.91$ | $19.64 \pm 0.94$ | $14.93 \pm 0.63$ | $43.73 \pm 1.16$ | $28.48 \pm 1.25$ |
| | G-retriever | $43.42 \pm 1.61$ | $37.33 \pm 1.19$ | $23.33 \pm 1.25$ | $18.96 \pm 0.86$ | $37.65 \pm 1.34$ | $34.59 \pm 1.02$ |
| | GRAG | $54.81 \pm 1.42$ | $34.52 \pm 0.88$ | $21.27 \pm 0.98$ | $19.99 \pm 0.75$ | $38.20 \pm 1.19$ | $36.55 \pm 1.32$ |
| FL + Local RAG | FLARE | $53.98 \pm 1.53$ | $25.25 \pm 1.27$ | $20.37 \pm 1.11$ | $11.88 \pm 0.64$ | $43.71 \pm 1.27$ | $28.67 \pm 0.97$ |
| | Dense RAG | $48.08 \pm 1.17$ | $34.03 \pm 1.34$ | $19.24 \pm 0.82$ | $9.63 \pm 0.68$ | $32.14 \pm 1.05$ | $31.01 \pm 1.16$ |
| Federated RAG | DGRAG | $49.54 \pm 1.31$ | $43.22 \pm 0.94$ | $20.18 \pm 0.93$ | $33.17 \pm 0.86$ | $42.13 \pm 1.12$ | $37.56 \pm 1.06$ |
| | FD-RAG | $54.47 \pm 1.24$ | $40.26 \pm 1.17$ | $36.22 \pm 1.08$ | $32.82 \pm 0.72$ | $41.94 \pm 0.97$ | $37.76 \pm 0.88$ |
| | **SynFedRAG (Ours)** | $\mathbf{64.77 \pm 1.38}$ | $\mathbf{44.70 \pm 1.24}$ | $\mathbf{37.24 \pm 1.16}$ | $\mathbf{34.53 \pm 0.86}$ | $\mathbf{43.81 \pm 1.21}$ | $\mathbf{41.08 \pm 1.22}$ |

### 4.5 PRIVACY LEAKAGE ANALYSIS

We assess membership leakage from representations available to the server and participating clients (Table 6); attack protocols are detailed in Appendix D.4. For target-client membership, condensation reduces leakage relative to the unshared full-graph reference, but uploaded condensed graphs still yield 67.1–72.9% AUC, exposing membership information to the server. The final synthetic graph yields near-chance AUCs of 49.5–51.1%, although its TPR at 1% FPR remains above the random baseline.

For membership in the union of other clients' corpora, accumulating snapshots produces no consistent increase across datasets, but raises the mean AUC on 2WikiMultiHopQA from 55.8% to 59.3%. Thus, the shared synthetic graph retains measurable membership signals under some evaluated conditions. Moreover, its low target-client AUC does not undo the server's earlier exposure to condensed graphs. Keeping raw documents local therefore limits direct data sharing but does not provide a formal privacy guarantee.

Table 6: Passage membership leakage from exposed graph representations. AUC and TPR at 1% FPR are reported in percentages. *Access:* $^{S}$server; $^{C}$participating clients. The full private graph is not shared. Group B targets membership in the union of other clients' corpora, using either the final received graph or all received snapshots.

| | **HotpotQA** | | **MuSiQue** | | **2WikiMultiHopQA** | |
|---|---|---|---|---|---|---|
| **Exposed representation** | AUC | TPR@1% | AUC | TPR@1% | AUC | TPR@1% |
| **A. Target-client membership** *(single snapshot)* | | | | | | |
| Full private graph (reference) | $100.0 \pm 0.0$ | 100.0 | $99.9 \pm 0.2$ | 100.0 | $100.0 \pm 0.1$ | 100.0 |
| Condensed client graph$^{S}$ | $72.9 \pm 2.5$ | 24.7 | $67.1 \pm 3.2$ | 20.6 | $67.1 \pm 3.7$ | 25.9 |
| Initial synthetic graph$^{S,C}$ | $50.7 \pm 3.2$ | 2.0 | $50.1 \pm 3.9$ | 2.6 | $51.0 \pm 3.4$ | 3.0 |
| Final synthetic graph$^{S,C}$ | $51.1 \pm 3.1$ | 2.1 | $49.5 \pm 3.6$ | 3.0 | $50.9 \pm 3.2$ | 1.9 |
| **B. Other-client membership** *(client attacker)* | | | | | | |
| Final graph snapshot$^{S,C}$ | $53.3 \pm 3.1$ | 2.7 | $51.7 \pm 3.5$ | 3.8 | $55.8 \pm 4.8$ | 3.0 |
| Accumulated snapshots$^{S,C}$ | $53.1 \pm 2.5$ | 2.3 | $52.6 \pm 5.3$ | 3.3 | $59.3 \pm 3.6$ | 5.3 |
| Random guess | 50.0 | 1.0 | 50.0 | 1.0 | 50.0 | 1.0 |

### 4.6 Ablation and In-Depth Analysis

We conduct ablations and diagnostic analyses to examine the design choices of SynFedRAG. Core component and objective ablations are summarized in Table 4. Additional analyses in the appendix include (1) synthetic-graph capacity, convergence, and communication efficiency (Appendix D.1), (2) comparison with federated LoRA baselines (Appendix D.2), (3) topic-matched question ownership (Appendix D.3), (4) passage-membership leakage (Appendix D.4), (5) the effect of regularization on synthetic-node diversity (Appendix D.5), and (6) qualitative examples and synthetic-node readout analysis (Appendix D.6).

## 5 Conclusion

This paper studied cross-client retrieval in federated GraphRAG, where relevant evidence is distributed across private corpora. We proposed SynFedRAG, which constructs a persistent shared synthetic graph from condensed client graphs and jointly refines this graph with a graph-guided prompt module using private QA supervision. Each client can then retrieve from both its private graph and the shared synthetic graph without contacting other clients at inference time. Experiments on three multi-hop QA benchmarks show improved overall QA under random and topic-skewed corpus partitions. The results therefore shift the main practical concern from preserving exact retrieval fidelity to the cost and exposure of the federated protocol: repeated prompt-module synchronization dominates offline communication, while client-condensed graphs account for the strongest measured membership leakage.

## AI USE STATEMENT

AI tools were used only to assist with language editing and improve the clarity of the manuscript. All scientific content, analyses, results, and conclusions were produced and verified by the authors.

## ETHICS STATEMENT

This work does not involve new human-subject data collection or deployment decisions. We use existing research datasets and follow their established research-use protocols.

## REPRODUCIBILITY STATEMENT

We provide the necessary methodological and experimental details to support reproducibility throughout the paper. The complete formulation of SynFedRAG, including graph construction and condensation, synthetic-graph initialization, retrieval, local refinement, and federated aggregation, is described in Appendix B. Experimental settings, dataset construction, implementation details, hyperparameters, baseline configurations, and evaluation protocols are provided in Appendix C.

## REFERENCES

Jieming Bian, Lei Wang, Letian Zhang, and Jie Xu. Lora-fair: Federated lora fine-tuning with aggregation and initialization refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 3737–3746, 2025.

Tianshi Che, Ji Liu, Yang Zhou, Jiaxiang Ren, Jiwen Zhou, Victor Sheng, Huaiyu Dai, and Dejing Dou. Federated learning of large language models with parameter-efficient prompt tuning and adaptive optimization. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7871–7888, 2023.

Yae Jee Cho, Luyang Liu, Zheng Xu, Aldi Fahrezi, and Gauri Joshi. Heterogeneous lora for federated fine-tuning of on-device foundation models. In *Proceedings of the 2024 conference on empirical methods in natural language processing*, pp. 12903–12913, 2024.

Tianhao Gao, Kai Yang, and Yiyang Li. Fd-rag: Federated dual-system retrieval-augmented generation. *arXiv preprint arXiv:2605.27432*, 2026a.

Yan Gao, Massimo R Scamarcia, Javier Fernandez-Marques, Mohammad Naseri, Chong Ng, Dimitris Stripelis, Zexi Li, Tao Shen, Jiamu Bai, Daoyuan Chen, et al. Flowertune: A cross-domain benchmark for federated fine-tuning of large language models. *Advances in Neural Information Processing Systems*, 38, 2026b.

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*, pp. 74–81, 2025.

Bernal J Gutiérrez, Yiheng Shu, Yu Gu, Michihiro Yasunaga, and Yu Su. Hipporag: Neurobiologically inspired long-term memory for large language models. *Advances in neural information processing systems*, 37:59532–59569, 2024.

Hangyu He, Xin Yuan, Kai Wu, Ren Ping Liu, and Wei Ni. pfedrag: A personalized federated retrieval-augmented generation system with depth-adaptive tiered embedding tuning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 14255–14268, 2025.

Xiaoxin He, Yijun Tian, Yifei Sun, Nitesh V Chawla, Thomas Laurent, Yann LeCun, Xavier Bresson, and Bryan Hooi. G-retriever: Retrieval-augmented generation for textual graph understanding and question answering. *Advances in Neural Information Processing Systems*, 37:132876–132907, 2024.

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 the 28th International Conference on Computational Linguistics*, pp. 6609–6625, 2020.

Yuntong Hu, Zhihan Lei, Zheng Zhang, Bo Pan, Chen Ling, and Liang Zhao. Grag: Graph retrieval-augmented generation. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 4145–4157, 2025.

Zhengbao Jiang, Frank F Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In *Proceedings of the 2023 conference on empirical methods in natural language processing*, pp. 7969–7992, 2023.

Wei Jin, Lingxiao Zhao, Shichang Zhang, Yozen Liu, Jiliang Tang, and Neil Shah. Graph condensation for graph neural networks. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=WLEx3Jo4QaB.

JungHyoun Kim, Soohyeong Kim, Seok Jun Hwang, Jeonghyeon Park, and Yong Suk Choi. Thrag: Topic-based hierarchical knowledge graphs for robust multi-hop reasoning in graph-based rag systems. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 37508–37531, 2026.

Kwun Hang Lau, Fangyuan Zhang, Boyu Ruan, Yingli Zhou, Qintian Guo, Ruiyuan Zhang, and Xiaofang Zhou. Breaking the static graph: Context-aware traversal for graph-based rag. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 5849–5863, 2026.

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

Linhao Luo, Zicheng Zhao, Reza Haffari, Dinh Phung, Chen Gong, and Shirui Pan. Gfm-rag: graph foundation model for retrieval augmented generation. *Advances in Neural Information Processing Systems*, 38:36371–36405, 2026.

Khang Nguyen, Nong Minh Hieu, Vinh Duc Nguyen, Nhat Ho, Stanley Osher, and Tan Minh Nguyen. Revisiting over-smoothing and over-squashing using ollivier-ricci curvature. In *International conference on machine learning*, pp. 25956–25979. PMLR, 2023.

Le-Tuan Nguyen, Minh-Duong Nguyen, Seon-Geun Jeong, Dung D Le, and Quoc-Viet Pham. Communication-efficient and accurate approach for aggregation in federated low-rank adaptation. *arXiv preprint arXiv:2509.26399*, 2025.

Raghav Singhal, Kaustubh Ponkshe, and Praneeth Vepakomma. Fedex-lora: Exact aggregation for federated and efficient fine-tuning of large language models. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1316–1336, 2025.

Harsh Trivedi, Niranjan Balasubramanian, Tushar Khot, and Ashish Sabharwal. Musique: Multihop questions via single-hop question composition. *Transactions of the Association for Computational Linguistics*, 10:539–554, 2022.

Juyuan Wang, Rongchen Zhao, Wei Wei, Yufeng Wang, Mo Yu, Jie Zhou, Jin Xu, and Liyan Xu. Comorag: A cognitive-inspired memory-organized rag for stateful long narrative reasoning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 33557–33565, 2026a.

Liang Wang, Haonan Chen, Nan Yang, Xiaolong Huang, Zhicheng Dou, and Furu Wei. Chain-of-retrieval augmented generation. *Advances in Neural Information Processing Systems*, 38:59888–59915, 2026b.

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

Zehua Wang, Zhaojin Zhang, Boyu Qiu, Xiaolong Weng, Ying Xiong, Buzhou Tang, and Min Zhang. Sr-rag: Verifiable multi-hop reasoning via on-the-fly symbolic graph construction. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 38588–38606, 2026c.

Ziyao Wang, Zheyu Shen, Yexiao He, Guoheng Sun, Hongyi Wang, Lingjuan Lyu, and Ang Li. Flora: Federated fine-tuning large language models with heterogeneous low-rank adaptations. *Advances in Neural Information Processing Systems*, 37:22513–22533, 2024b.

Zhepei Wei, Wei-Lin Chen, and Yu Meng. Instructrag: Instructing retrieval-augmented generation via self-synthesized rationales. In *International Conference on Learning Representations*, volume 2025, pp. 82731–82754, 2025.

Feijie Wu, Zitao Li, Yaliang Li, Bolin Ding, and Jing Gao. Fedbiot: Llm local fine-tuning in federated learning without full model. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 3345–3355, 2024.

Yiyuan Yang, Guodong Long, Tao Shen, Jing Jiang, and Michael Blumenstein. Dual-personalizing adapter for federated foundation models. *Advances in Neural Information Processing Systems*, 37:39409–39433, 2024.

Zhibang Yang, Xinke Jiang, Rihong Qiu, Ruiqing Li, Yihang Zhang, Yue Fang, Yongxin Xu, Hongxin Ding, Xu Chu, Junfeng Zhao, et al. Dfams: Dynamic-flow guided federated alignment based multi-prototype search. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24728–24752, 2026.

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 the 2018 conference on empirical methods in natural language processing*, pp. 2369–2380, 2018.

Yue Yu, Wei Ping, Zihan Liu, Boxin Wang, Jiaxuan You, Chao Zhang, Mohammad Shoeybi, and Bryan Catanzaro. Rankrag: Unifying context ranking with retrieval-augmented generation in llms. *Advances in Neural Information Processing Systems*, 37:121156–121184, 2024.

Jianyi Zhang, Saeed Vahidian, Martin Kuo, Chunyuan Li, Ruiyi Zhang, Tong Yu, Guoyin Wang, and Yiran Chen. Towards building the federatedgpt: Federated instruction tuning. In *ICASSP 2024-2024 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 6915–6919. IEEE, 2024.

Wenqing Zhou, Yuxuan Yan, and Qianqian Yang. Dgrag: Distributed graph-based retrieval-augmented generation in edge-cloud systems. *arXiv preprint arXiv:2505.19847*, 2025.

Luyao Zhuang, Shengyuan Chen, Yilin Xiao, Huachi Zhou, Yujing Zhang, Hao Chen, Qinggang Zhang, and Xiao Huang. Linearrag: Linear graph retrieval augmented generation on large-scale corpora. *arXiv preprint arXiv:2510.10114*, 2025.

# A Related Work

SynFedRAG lies at the intersection of federated LLM adaptation, retrieval-augmented generation, and decentralized knowledge sharing. Federated LLM methods primarily exchange parameter-efficient model updates, whereas RAG and GraphRAG provide non-parametric knowledge through external textual or graph-structured memories. Federated RAG combines these directions by enabling retrieval over distributed private corpora. We review these lines of work and clarify the design space addressed by SynFedRAG.

## A.1 Federated LLM Adaptation

Recent work on federated LLM adaptation has largely focused on enabling efficient collaboration under limited client computation and communication resources. Federated prompt tuning and adaptive optimization have been explored for LLM adaptation under decentralized data Che et al. (2023), while low-rank adaptation has been studied for heterogeneous and communication-efficient federated fine-tuning Wang et al. (2024b); Cho et al. (2024); Singhal et al. (2025); Nguyen et al. (2025); Bian et al. (2025). Other works investigate federated instruction tuning, personalized adapters, on-device foundation model adaptation, and federated LLM benchmarks Zhang et al. (2024); Yang et al. (2024); Wu et al. (2024); Gao et al. (2026b).

Although these methods reduce the computational and communication costs of federated LLM adaptation, they primarily facilitate knowledge transfer through model-side parameters, such as prompts, adapters, and low-rank factors. However, the external knowledge required for retrieval remains distributed across private and continuously evolving client corpora, making model updates alone insufficient for knowledge-intensive question answering. This raises an important question: how can retrieval knowledge be effectively transferred among clients? SynFedRAG addresses this gap by transferring retrieval knowledge through a shared synthetic graph.

## A.2 Knowledge-Supported LLMs: RAG and GraphRAG

Retrieval-augmented generation (RAG) grounds language-model outputs in external evidence, alleviating the limitations of static parametric knowledge (Lewis et al., 2020). Subsequent work improves how evidence is selected and incorporated. FLARE performs active retrieval during generation by using anticipated future content to determine when and what to retrieve (Jiang et al., 2023), while ranking and multi-step methods improve context selection and evidence composition (Yu et al., 2024; Wei et al., 2025; Wang et al., 2026b). ComoRAG further introduces a stateful retrieval process that iteratively acquires and consolidates evidence in a dynamic memory workspace for long-context reasoning (Wang et al., 2026a). These methods demonstrate the importance of adaptive evidence acquisition for knowledge-intensive and multi-step generation.

GraphRAG extends RAG by organizing evidence into graph-structured memories. Representative methods support relational reasoning through memory-inspired graph retrieval, question-relevant subgraph selection, and graph-guided generation (Gutiérrez et al., 2024; He et al., 2024; Hu et al., 2025). Other approaches improve multi-hop retrieval through relation-free entity–sentence–passage graphs, hierarchical knowledge organization, graph foundation models, and context-aware traversal (Zhuang et al., 2025; Luo et al., 2026; Kim et al., 2026; Lau et al., 2026; Wang et al., 2026c). These structures facilitate relational and multi-hop evidence discovery beyond independent text-chunk retrieval. However, existing RAG and GraphRAG systems typically assume that the retrieval corpus, embedding index, or graph memory is centrally available. This assumption is restrictive in federated or multi-organization settings, where documents are distributed across private clients and cannot be directly pooled.

## A.3 Federated Retrieval-Augmented Generation

A growing line of work studies retrieval-augmented generation under federated or decentralized settings. Existing approaches formulate the problem as federated search for RAG, efficient federated search mechanisms, personalized federated RAG systems and prototype-based alignment for decentralized retrieval scenarios Wang et al. (2024a); Guerraoui et al. (2025); He et al. (2025); Yang et al.

(2026). These studies highlight an important limitation of local-only retrieval: a client may lack the evidence needed to answer a query, even when such evidence exists in another client corpus.

Recent systems additionally exploit structured knowledge: DGRAG Zhou et al. (2025) exchanges summaries of local knowledge graphs, but primarily uses them to identify relevant edge repositories and still obtains cross-client evidence through query-time remote retrieval. FD-RAG Gao et al. (2026a) distills local semantic hypergraphs into federated QA memories, but its global knowledge store aggregates memory items rather than learning a graph-structured retrieval space whose representations and topology are jointly optimized. These works establish the need for cross-client retrieval, but leave open how graph-structured knowledge can be transferred into a persistent retrieval resource that is locally available to each client.

**Positioning.** Taken together, prior works reveal a specific gap. Federated LLM adaptation transfers model-side knowledge but does not make evidence stored in private client corpora directly retrievable; RAG and GraphRAG provide access to such non-parametric evidence but conventionally assume a centrally available corpus or graph memory; and existing federated RAG systems relax this assumption through distributed search, retriever adaptation, graph-summary exchange, or memory distillation, without learning a shared graph retrieval space that is both persistent and locally available to every client. SynFedRAG targets this missing capability by transforming retrieval-preserving summaries of tri-graphs into a synthetic graph. The graph itself serves as the cross-client retrieval resource: it can be queried locally together with each client's private graph, refined through federated QA supervision, and used without centralizing complete graphs or contacting other clients at inference time.

Table 7: Main notation used in SynFedRAG. Round and local-step indices are omitted where clear.

| Notation | Description |
|---|---|
| **Federated setting and graphs** | |
| $M,\ m$ | Number of clients and client index. |
| $\mathcal{D}_m,\ \mathcal{Q}_m,\ n_m$ | Private corpus, QA dataset, and number of QA pairs. |
| $G_m = (V_m, E_m, \mathbf{X}_m)$ | Private graph: nodes, edges, and node features. |
| $V_m^E,\ V_m^S,\ V_m^P$ | Entity, sentence, and passage node sets. |
| $\widetilde{G}_m$ | Client-condensed graph uploaded in Phase I. |
| $\mathcal{M}_{\text{syn}}$ | Shared synthetic graph, parameterized by $\mathbf{X}_{\text{syn}}$ and $\boldsymbol{\theta}_{\text{PGE}}$. |
| $K_g,\ \mathbf{X}_{\text{syn}},\ \mathbf{A}_{\text{syn}}$ | Synthetic-node count, node features, and induced adjacency. |
| **Graph representations and retrieval** | |
| $f_{\text{GNN}},\ g_{\text{proj}}$ | Graph encoder and graph-to-LLM projector. |
| $W$ | Prompt-module parameters: $\{\boldsymbol{\theta}_{\text{GNN}}, \boldsymbol{\theta}_{\text{proj}}\}$. |
| $d_x,\ d_g,\ d_{\text{LLM}}$ | Node-feature, GNN-output, and LLM-embedding dimensions. |
| $z_m,\ z_{\text{syn}}$ | Projected graph-level representations for alignment. |
| $\mathcal{C}_m^{\text{loc}}(q),\ \mathcal{C}_m^{\text{syn}}(q)$ | Local and synthetic graph contexts for question $q$. |
| $e_q,\ \alpha_i(q)$ | Question embedding and synthetic-node retrieval weight. |
| $\mathbf{H}_{\text{syn}},\ h_i^{\text{syn}}$ | Synthetic GNN outputs and the representation of node $i$. |
| $z_q^{\text{loc}},\ z_q^{\text{syn}}$ | Local and synthetic soft graph tokens for question $q$. |
| **Federated refinement** | |
| $r,\ \mathcal{S}_r$ | Communication-round index and participating clients. |
| $W^{(r)},\ W_m^{(r)}$ | Broadcast and locally adapted prompt-module parameters. |
| $\mathcal{M}_{\text{syn}}^{(r)}$ | Broadcast graph. |
| $K_{\text{mem}}$ | Number of local graph-refinement steps per round. |
| $\Delta_m^{(r)},\ \Delta^{(r)}$ | Client graph-parameter update and its weighted aggregate. |
| **Training objectives** | |
| $\mathcal{L}_{\text{cond}}^{(m)}$ | Client-side graph-condensation objective. |
| $\mathcal{L}_{\text{align}}$ | graph alignment Loss. |
| $\mathcal{L}_{\text{QA}}^{(m)}$ | Answer-token negative log-likelihood. |
| $\mathcal{L}_{\text{GM}}^{(m)}$ | QA-gradient matching between local and synthetic contexts. |
| $\mathcal{L}_{\text{mem}}^{(m)}$ | Combined local synthetic-graph refinement objective. |
| $\mathcal{L}_{\text{reg}}$ | Weighted cosine-similarity penalty on synthetic-node features. |

# B Detailed Algorithm

## B.1 Notation

Table 7 summarizes the main notation. Algorithm 1 presents both phases of SynFedRAG and specifies the information exchanged between the server and clients. The following subsections detail graph condensation, synthetic topology, and client-side refinement.

Throughout the algorithm, the LLM parameters remain frozen. Synthetic-graph updates act on $\mathbf{X}_{\text{syn}}$ and $\boldsymbol{\theta}_{\text{PGE}}$; the adjacency is computed from these parameters and is not transmitted separately. Dur-

**Algorithm 1** SynFedRAG: initialization and federated refinement

**Require:** Private corpora $\{\mathcal{D}_m\}_{m=1}^{M}$; private QA sets $\{\mathcal{Q}_m\}_{m=1}^{M}$; synthetic-node budget $K_g$; rounds $T$; local graph-refinement steps $K_{\mathrm{mem}}$

**Ensure:** Shared prompt module $W^{(T)}$ and synthetic graph $\mathcal{M}_{\mathrm{syn}}^{(T)}$

**Phase I: Synthetic graph initialization**

**for** each client $m$ **do**
  Construct the private tri-graph $G_m$ from $\mathcal{D}_m$.
  Select entity anchors, expand motifs, and construct $\widetilde{G}_m$.
  Refine condensed graph by minimizing $\mathcal{L}_{\mathrm{ret}}^{(m)} + \lambda_{\mathrm{rep}}\mathcal{L}_{\mathrm{rep}}^{(m)} + \lambda_{\mathrm{div}}\mathcal{L}_{\mathrm{div}}^{(m)}$.
  **Client** $m \to$ **server:** upload $\widetilde{G}_m$ once.
**end for**
**Server:** initialize the prompt module $W^{(0)}$ and synthetic state $\mathcal{M}_{\mathrm{syn}} = \{\mathbf{X}_{\mathrm{syn}}, \boldsymbol{\theta}_{\mathrm{PGE}}\}$ with $K_g$ nodes.
Hold $W^{(0)}$ and $\{\widetilde{G}_m\}_{m=1}^{M}$ fixed; optimize $\mathcal{M}_{\mathrm{syn}}$ using $\mathcal{L}_{\mathrm{align}}^{\mathrm{init}} + \mathcal{L}_{\mathrm{reg}}$ to obtain $\mathcal{M}_{\mathrm{syn}}^{(0)}$.

**Phase II: Federated synthetic graph refinement**

**for** $r = 0, \ldots, T-1$ **do**
  Select participating clients $\mathcal{S}_r$.
  **Server** $\to$ **each** $m \in \mathcal{S}_r$**:** download $W^{(r)}$ and $\mathcal{M}_{\mathrm{syn}}^{(r)} = \{\mathbf{X}_{\mathrm{syn}}^{(r)}, \boldsymbol{\theta}_{\mathrm{PGE}}^{(r)}\}$.
  **for** each participating client $m \in \mathcal{S}_r$ **do**
    Initialize $W_m^{(r)} \leftarrow W^{(r)}$ and $\mathcal{M}_{\mathrm{syn},m}^{(r)} \leftarrow \mathcal{M}_{\mathrm{syn}}^{(r)}$.
    Retrieve local evidence and compute question-conditioned synthetic contexts.
    Adapt $W_m^{(r)}$ using $\mathcal{L}_{\mathrm{QA}}^{(m)}$, holding the synthetic state fixed.
    Hold the adapted values of $W_m^{(r)}$ fixed during graph refinement.
    **for** $k = 0, \ldots, K_{\mathrm{mem}} - 1$ **do**
      Compute $\mathcal{L}_{\mathrm{mem}}^{(m)} = \mathcal{L}_{\mathrm{QA}}^{(m)} + \lambda_{\mathrm{gm}}\mathcal{L}_{\mathrm{GM}}^{(m)} + \lambda_{\mathrm{align}}\mathcal{L}_{\mathrm{align}}^{(m)}$.
      Update $\mathcal{M}_{\mathrm{syn},m}^{(r,k+1)} \leftarrow \mathcal{M}_{\mathrm{syn},m}^{(r,k)} - \eta_{\mathrm{syn}}\nabla_{\mathcal{M}_{\mathrm{syn},m}^{(r,k)}}\mathcal{L}_{\mathrm{mem}}^{(m)}$.
    **end for**
    Form $\Delta_m^{(r)} \leftarrow \mathcal{M}_{\mathrm{syn},m}^{(r,K_{\mathrm{mem}})} - \mathcal{M}_{\mathrm{syn}}^{(r)}$.
    **Client** $m \to$ **server:** upload $W_m^{(r)}$ and $\Delta_m^{(r)}$.
  **end for**
  **Server:** set $\omega_m^{(r)} = n_m / \sum_{j \in \mathcal{S}_r} n_j$.
  Aggregate $W^{(r+1)} \leftarrow \sum_{m \in \mathcal{S}_r} \omega_m^{(r)} W_m^{(r)}$ and $\Delta^{(r)} \leftarrow \sum_{m \in \mathcal{S}_r} \omega_m^{(r)} \Delta_m^{(r)}$.
  Update $\mathcal{M}_{\mathrm{syn}}^{(r+1)} \leftarrow \mathcal{M}_{\mathrm{syn}}^{(r)} + \Delta^{(r)} - \eta_{\mathrm{reg}}\nabla_{\mathcal{M}_{\mathrm{syn}}}\mathcal{L}_{\mathrm{reg}}$.
**end for**

ing graph refinement, prompt-module parameter values remain fixed, while the derivatives required for gradient matching and synthetic-state optimization are retained.

### B.2 Retrieval-Preserving Graph Condensation

This appendix provides the construction details and the explicit forms of the condensation losses introduced in Sec. 2.2.1.

#### B.2.1 Anchor Selection and Motif Expansion

**Anchor budget.** For client $m$, we select $K_m^E$ entity anchors, where

$$K_m^E = \max\left(K_{\min}, \left\lceil \rho_E |V_m^E| \right\rceil\right), \tag{14}$$

where $V_m^E$ denotes the entity nodes in the local tri-graph and $\rho_E$ controls the entity-anchor budget.

For each entity $e \in V_m^E$, we define

$$s(e) = \log(1 + \deg_S(e)) + \log(1 + \deg_P(e)) + \lambda_{\mathrm{idf}}\,\mathrm{IDF}_m(e) + \lambda_{\mathrm{pr}}\,\mathrm{PR}_m(e), \tag{15}$$

where

$$\mathrm{IDF}_m(e) = \log \frac{|V_m^P|}{1 + \deg_P(e)}, \tag{16}$$

with passage nodes serving as documents and $\deg_P(e)$ giving the number of adjacent passages of $e$ in the local entity–passage graph. $\mathrm{PR}_m(e)$ denotes the PageRank score computed on the client-local tri-graph $G_m$. To reduce redundancy, we select anchors greedily using maximal marginal relevance. Let $\widetilde{V}_{m,\mathrm{anc}} \subseteq V_m^E$ denote the set of entity anchors selected so far, initialized as the empty set. At each step, we select

$$e^\star = \arg\max_{e \notin \widetilde{V}_{m,\mathrm{anc}}} \left[ s(e) - \lambda_{\mathrm{mmr}} \max_{a \in \widetilde{V}_{m,\mathrm{anc}}} \cos(x_e, x_a) \right]. \tag{17}$$

We repeat this selection until $K_m^E$ anchors have been obtained.

Here, Top-$b$ selects the $\min(b, |\mathcal{S}|)$ highest-scoring elements from a candidate set $\mathcal{S}$. For each anchor $a$, $\mathcal{N}_S(a)$ and $\mathcal{N}_P(a)$ denote its neighboring sentence and passage nodes, respectively; $b_S$ and $b_P$ specify the maximum numbers of these nodes retained per anchor.

$$H(a) = \{a\} \cup \text{Top-}b_S(N_S(a)) \cup \text{Top-}b_P(N_P(a)), \tag{18}$$

and

$$\widetilde{V}_m = \bigcup_{a \in \widetilde{V}_{m,\mathrm{anc}}} H(a). \tag{19}$$

### B.2.2 Neighborhood Condensation and Topology Reconstruction

For each $\tilde{v} \in \widetilde{V}_m$ and hop $h \in \{0, 1, 2\}$, we retain up to $b_h$ nodes from its $h$-hop neighborhood $\mathcal{N}^{(h)}(\tilde{v})$, ranked by cosine similarity to $x_{\tilde{v}}$:

$$\mathcal{S}_{\tilde{v}}^{(h)} = \text{Top-}b_h \left\{ u \in \mathcal{N}^{(h)}(\widetilde{v}) : \cos(x_{\widetilde{v}}, x_u) \right\}, \tag{20}$$

with normalized weights

$$\alpha_{\widetilde{v}u}^{(h)} = \frac{\exp(\cos(x_{\widetilde{v}}, x_u)/\tau_n)}{\sum_{u' \in \mathcal{S}_{\widetilde{v}}^{(h)}} \exp(\cos(x_{\widetilde{v}}, x_{u'})/\tau_n)}. \tag{21}$$

For each selected node $u$, let $c_u^k \in \mathbb{R}^{d_x}$ denote the frozen text-encoder embedding of its $k$-th local text chunk. Let $\mathcal{K}_{\widetilde{v}}^\star$ denote the top-$b_c$ chunk embeddings selected from the neighborhoods according to

$$\ell_{\widetilde{v},u,k} = \frac{(c_u^k)^\top x_{\widetilde{v}}}{\sqrt{d_x}} + \log\left(\alpha_{\widetilde{v}u}^{(h(u))} + \epsilon\right). \tag{22}$$

Their normalized weights are

$$\beta_{\widetilde{v},u,k} = \frac{\exp(\ell_{\widetilde{v},u,k}/\tau_c)}{\sum_{c_{u'}^{k'} \in \mathcal{K}_{\widetilde{v}}^\star} \exp(\ell_{\widetilde{v},u',k'}/\tau_c)}, \tag{23}$$

yielding the neighborhood-condensed representation

$$\widetilde{t}_{\widetilde{v}} = \sum_{c_u^k \in \mathcal{K}_{\widetilde{v}}^\star} \beta_{\widetilde{v},u,k} c_u^k. \tag{24}$$

Let $\phi_m(\widetilde{v}) \in V_m$ denote the original node associated with $\widetilde{v}$. Its initial condensed feature is

$$\widetilde{x}_{\widetilde{v}}^{(0)} = \mathrm{Normalize}_2\left((1-\rho) x_{\phi_m(\widetilde{v})} + \rho \widetilde{t}_{\widetilde{v}}\right). \tag{25}$$

The condensed topology retains inherited edges and adds semantic nearest-neighbor edges. For $\widetilde{v}_i, \widetilde{v}_j \in \widetilde{V}_m$, we use

$$S_{ij} = \cos\left(\widetilde{x}_{\widetilde{v}_i}^{(0)}, \widetilde{x}_{\widetilde{v}_j}^{(0)}\right) + \cos\left(\widetilde{t}_{\widetilde{v}_i}, \widetilde{t}_{\widetilde{v}_j}\right), \tag{26}$$

and connect each node to its top-$k_{\mathrm{topo}}$ neighbors according to $S_{ij}$.

### B.2.3 CONDENSATION LOSSES

**Retrieval preservation.** The condensed graph should preserve the query-dependent passage preferences of the full private graph while remaining differentiable with respect to its learnable node features. For each training question $q$, we therefore use the passage-ranking distribution produced by LinearRAG on the full private graph $G_m$ as a fixed teacher, denoted by $r_{G_m}(p \mid q)$ for $p \in V_m^P$. Since $G_m$ remains fixed during condensation, these teacher distributions are precomputed and cached in our implementation, and no gradients are propagated through the LinearRAG retrieval procedure.

For the condensed graph, directly differentiating through the full LinearRAG ranking pipeline is unnecessary. Instead, we define a differentiable query-conditioned surrogate over condensed passage nodes. Let $e_q = \mathrm{Enc}_{\mathrm{text}}(q)$ be the question embedding produced by the same frozen text encoder used to construct graph-node features. For $\widetilde{p} \in \widetilde{V}_m^P$, we define

$$r_{\widetilde{G}_m}(\widetilde{p} \mid q) = \frac{\exp(\cos(e_q, \widetilde{x}_{\widetilde{p}})/\tau_{\mathrm{ret}})}{\sum_{\widetilde{p}' \in \widetilde{V}_m^P} \exp(\cos(e_q, \widetilde{x}_{\widetilde{p}'})/\tau_{\mathrm{ret}})}. \tag{27}$$

Because $r_{G_m}(\cdot \mid q)$ and $r_{\widetilde{G}_m}(\cdot \mid q)$ are defined over different passage sets, we map the condensed distribution back to the original passage space. For each condensed passage $\widetilde{p}$, we define its soft coverage over the full passage set as

$$C_m(p \mid \widetilde{p}) = \frac{\exp(\cos(x_p, \widetilde{x}_{\widetilde{p}})/\tau_a)}{\sum_{p' \in V_m^P} \exp(\cos(x_{p'}, \widetilde{x}_{\widetilde{p}})/\tau_a)}. \tag{28}$$

The condensed retrieval distribution is then lifted to the original passage space:

$$\widehat{r}_{\widetilde{G}_m}(p \mid q) = \sum_{\widetilde{p} \in \widetilde{V}_m^P} C_m(p \mid \widetilde{p})\, r_{\widetilde{G}_m}(\widetilde{p} \mid q). \tag{29}$$

We optimize the condensed passage representations by matching this lifted distribution to the fixed LinearRAG teacher:

$$\mathcal{L}_{\mathrm{ret}}^{(m)} = \mathbb{E}_{q \sim Q_m}\left[ D_{\mathrm{KL}}\left( \mathrm{sg}[r_{G_m}(\cdot \mid q)] \,\middle\|\, \widehat{r}_{\widetilde{G}_m}(\cdot \mid q) \right) \right], \tag{30}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. Thus, LinearRAG provides the full-graph retrieval target, while the condensed branch remains fully differentiable with respect to $\widetilde{X}_m$. In particular, gradients from $\mathcal{L}_{\mathrm{ret}}^{(m)}$ affect the condensed passage-node features through the cosine-softmax surrogate and soft-coverage mapping.

**Representation preservation.** Retrieval matching alone under-constrains the condensation problem, since multiple condensed graphs can reproduce similar retrieval behavior on the observed queries while preserving different amounts of the underlying graph information. This matters in SynFedRAG because the condensed graph is subsequently encoded to provide the client-side representation signal for synthetic-graph initialization. We therefore complement retrieval matching with $\mathcal{L}_{\mathrm{rep}}^{(m)}$, which preserves contextual representation geometry under a fixed reference GNN whose parameters $\theta_{\mathrm{GNN}}^{\mathrm{ref}}$ are randomly initialized with a fixed seed and frozen throughout condensation.

$$H_m = f_{\mathrm{GNN}}(G_m; \boldsymbol{\theta}_{\mathrm{GNN}}^{\mathrm{ref}}), \qquad \widetilde{H}_m = f_{\mathrm{GNN}}(\widetilde{G}_m; \boldsymbol{\theta}_{\mathrm{GNN}}^{\mathrm{ref}}), \tag{31}$$

we construct a type-constrained soft assignment

$$\Pi_m(v, \widetilde{u}) = \begin{cases} \dfrac{\exp\Big(\cos(\mathrm{sg}(h_v), \widetilde{h}_{\widetilde{u}})/\tau_\pi\Big)}{\sum_{\widetilde{w} \in \widetilde{V}_m^{\mathrm{type}(v)}} \exp\Big(\cos(\mathrm{sg}(h_v), \widetilde{h}_{\widetilde{w}})/\tau_\pi\Big)}, & \tau(\widetilde{u}) = \mathrm{type}(v), \\ 0, & \text{otherwise}, \end{cases} \tag{32}$$

where $\mathrm{type}(v) \in \{E, S, P\}$ denotes node type. For a type-stratified subset $\mathcal{B}_m \subseteq V_m$,

$$\mathcal{L}_{\mathrm{rep}}^{(m)} = \frac{1}{|\mathcal{B}_m|} \sum_{v \in \mathcal{B}_m} \left\| \mathrm{sg}(h_v) - \sum_{\widetilde{u} \in \widetilde{V}_m} \Pi_m(v, \widetilde{u}) \widetilde{h}_{\widetilde{u}} \right\|_2^2. \tag{33}$$

**Condensed-node diversity.** While the MMR criterion in Eq. (17) reduces redundancy during anchor selection, subsequent feature refinement can still make distinct condensed nodes converge to similar representations. This wastes effective capacity under a fixed node budget. We therefore introduce $L_{\text{div}}^{(m)}$ to discourage excessive overlap among same-type condensed nodes. Let $K_{m,t} = |\widetilde{V}_m^t|$ for $t \in \{E, S, P\}$. We define

$$\mathcal{L}_{\text{div}}^{(m)} = \sum_{\substack{t \in \{E,S,P\} \\ K_{m,t} > 1}} \frac{1}{K_{m,t}(K_{m,t}-1)} \sum_{\substack{\widetilde{u},\widetilde{v} \in \widetilde{V}_m^t \\ \widetilde{u} \neq \widetilde{v}}} \left[\cos(\widetilde{x}_{\widetilde{u}}, \widetilde{x}_{\widetilde{v}}) - \delta_t\right]_+ , \tag{34}$$

where $[z]_+ = \max(0, z)$.

### B.3 Synthetic Graph

This appendix provides the topology parameterization and regularizer for the synthetic graph introduced in Sec. 2.2.2.

#### B.3.1 Parameterized Synthetic Topology

Recall that

$$\mathcal{M}_{\text{syn}} = \{\mathbf{X}_{\text{syn}}, \boldsymbol{\theta}_{\text{PGE}}\}, \qquad \mathbf{X}_{\text{syn}} \in \mathbb{R}^{K_g \times d_x}.$$

Rather than maintaining an independent adjacency matrix, the topology is induced from the synthetic-node representations by a parameterized graph estimator. For synthetic nodes $i$ and $j$, we compute the soft edge weight

$$a_{ij} = \sigma\left(\text{PGE}_{\boldsymbol{\theta}_{\text{PGE}}}\left([\mathbf{x}_i^{\text{syn}}, \mathbf{x}_j^{\text{syn}}]\right)\right), \tag{35}$$

where $\mathbf{x}_i^{\text{syn}}$ and $\mathbf{x}_j^{\text{syn}}$ denote the corresponding rows of $\mathbf{X}_{\text{syn}}$. Collecting $\{a_{ij}\}$ yields the soft adjacency $\mathbf{A}_{\text{syn}}$, which is used by graph operations applied to $\mathcal{M}_{\text{syn}}$ during training and inference. Thus, the synthetic topology is determined by $\mathbf{X}_{\text{syn}}$ and $\boldsymbol{\theta}_{\text{PGE}}$ rather than optimized as a separate graph variable.

#### B.3.2 Synthetic-Graph Regularization

Graph-level alignment does not directly constrain the distinctiveness of individual synthetic nodes. The regularizer defined in Section 2.2.2 penalizes pairwise cosine similarities above $\delta_{\text{syn}}$ among the learnable features $\mathbf{X}_{\text{syn}}$. It depends only on these features and can therefore be evaluated directly on the server during both initialization and federated refinement.

### B.4 Client-Side Synthetic Graph Refinement

This appendix provides the detailed formulations of the client-local alignment and gradient-matching objectives used to refine the synthetic graph in Sec. 2.3.1. All quantities in this appendix are computed locally at client $m$.

#### B.4.1 Local Graph Retrieval

**Passage retrieval.** Given a question $q$, client $m$ retrieves relevant passages from its private graph $G_m$ following LinearRAG Zhuang et al. (2025). First, query-linked entities activate relevant intermediate entities through query-guided propagation on the entity–sentence subgraph. Second, personalized PageRank on the entity–passage subgraph ranks passages using the activated entities and query–passage similarity. We select the highest-ranked passages within the retrieval budget.

**Evidence subgraph construction.** We construct $C_m^{\text{loc}}(q)$ as the subgraph induced by the retrieved passage nodes and their one-hop entity neighbors in $G_m$. It contains the retrieved passages, the entities they mention, and the passage–entity edges connecting them. We use unit edge weights and omit edge-type labels when encoding this subgraph with the GNN. The retrieved passage text provides $T_{\text{desc}}$ in Eq. (8).

### B.4.2 Differentiable Synthetic Context Retrieval

We use soft retrieval to read from the synthetic graph through question-conditioned attention over all $K_g$ nodes, producing a single soft token rather than selecting discrete passages. During synthetic-graph refinement, synthetic retrieval should remain conditioned directly on the input question rather than on the locally retrieved context. This is important because the shared synthetic graph is intended to provide complementary evidence even when the querying client's private graph does not contain sufficient relevant information. Let $e_q = \mathrm{Enc}_{\mathrm{text}}(q) \in \mathbb{R}^{d_x}$ denote the question embedding produced by the same text encoder used to construct graph-node features, and let $x_i^{\mathrm{syn}} \in \mathbb{R}^{d_x}$ denote the feature representation of synthetic node $i$. We compute a differentiable query-conditioned retrieval weight over the synthetic nodes as

$$\alpha_i(q) = \frac{\exp\left(\cos\left(e_q, x_i^{\mathrm{syn}}\right)/\tau\right)}{\sum_{j=1}^{K_g} \exp\left(\cos\left(e_q, x_j^{\mathrm{syn}}\right)/\tau\right)}, \tag{36}$$

where $\tau$ is a temperature parameter. To incorporate the learned synthetic graph structure, we first compute graph-contextualized synthetic-node representations

$$\mathbf{H}_{\mathrm{syn}} = f_{\mathrm{GNN}}\left(\mathbf{X^{(r)}}_{\mathrm{syn}}, \mathbf{A^{(r)}}_{\mathrm{syn}}; \boldsymbol{\theta}^{(r)}_{\mathrm{GNN},m}\right), \tag{37}$$

where $h_i^{\mathrm{syn}}$ denotes the $i$-th row of $\mathbf{H}_{\mathrm{syn}}$. The query-conditioned synthetic context is then obtained by weighted aggregation:

$$z_q^{\mathrm{syn}} = g_{\mathrm{proj}}\left(\sum_{i=1}^{K_g} \alpha_i(q) h_i^{\mathrm{syn}}; \boldsymbol{\theta}^{(r)}_{\mathrm{proj},m}\right). \tag{38}$$

This formulation keeps synthetic retrieval directly conditioned on the question and independent of the quality of local retrieval, while preserving a differentiable path from the QA objective to the synthetic graph.

### B.4.3 Client-Local Representation Alignment

In Phase I, the server aligns the initial synthetic graph with the uploaded client-condensed graphs. During Phase II, the corresponding alignment is computed locally so that client-condensed representations need not be transmitted again.

Using the locally adapted graph-guided prompt module $W_m^{(r)}$, client $m$ encodes its condensed graph and current synthetic-graph copy using the same graph-level mapping as in Phase I:

$$z_m^{(r)} = g_{\mathrm{proj}}\left(\mathrm{Pool}\left[f_{\mathrm{GNN}}\left(\widetilde{G}_m; \boldsymbol{\theta}^{(r)}_{\mathrm{GNN},m}\right)\right]; \boldsymbol{\theta}^{(r)}_{\mathrm{proj},m}\right), \tag{39}$$

$$z_{\mathrm{syn},m}^{(r,k)} = g_{\mathrm{proj}}\left(\mathrm{Pool}\left[f_{\mathrm{GNN}}\left(\mathcal{M}_{\mathrm{syn},m}^{(r,k)}; \boldsymbol{\theta}^{(r)}_{\mathrm{GNN},m}\right)\right]; \boldsymbol{\theta}^{(r)}_{\mathrm{proj},m}\right). \tag{40}$$

We then define the client-local alignment objective as

$$\mathcal{L}_{\mathrm{align}}^{(m)} = 1 - \cos\left(z_m^{(r)}, z_{\mathrm{syn},m}^{(r,k)}\right). \tag{41}$$

This term preserves consistency between the evolving synthetic graph and the client-specific graph information transferred during Phase I.

### B.4.4 QA-Gradient Matching

Representation alignment alone encourages the synthetic graph to preserve client graph information, but does not ensure that synthetic context induces the same QA-relevant behavior as private evidence. We therefore match the graph-guided prompt module gradients induced by the two context sources.

For a private QA mini-batch $\mathcal{B}_m$, let

$$g_m^{\mathrm{loc}} = \mathrm{sg}\left[\nabla_W \mathcal{L}_{\mathrm{QA}}^{(m)}\left(\mathcal{B}_m; \mathcal{C}_m^{\mathrm{loc}}, W_m^{(r)}\right)\right], \tag{42}$$

where $\mathrm{sg}(\cdot)$ denotes stop-gradient. Similarly, the gradient induced by the current synthetic-graph copy is

$$g_m^{\mathrm{syn}} = \nabla_W \mathcal{L}_{\mathrm{QA}}^{(m)} \left( \mathcal{B}_m; \mathcal{C}_m^{\mathrm{syn}} \left( \mathcal{M}_{\mathrm{syn},m}^{(r,k)} \right), W_m^{(r)} \right). \tag{43}$$

We minimize their cosine distance:

$$\mathcal{L}_{\mathrm{GM}}^{(m)} = 1 - \frac{\left\langle g_m^{\mathrm{loc}}, g_m^{\mathrm{syn}} \right\rangle}{\|g_m^{\mathrm{loc}}\|_2 \, \|g_m^{\mathrm{syn}}\|_2 + \epsilon}. \tag{44}$$

The local-evidence gradient is treated as a fixed target, while the synthetic branch remains differentiable with respect to $\mathcal{M}_{\mathrm{syn},m}^{(r,k)}$. Consequently, minimizing $\mathcal{L}_{\mathrm{GM}}^{(m)}$ encourages the synthetic graph to produce QA-relevant adaptation directions similar to those induced by private retrieved evidence.

#### B.4.5 Local Graph Objective

Combining QA supervision, representation alignment, and gradient matching gives the local synthetic-graph objective

$$\mathcal{L}_{\mathrm{mem}}^{(m)} = \mathcal{L}_{\mathrm{QA}}^{(m)} + \lambda_{\mathrm{gm}} \mathcal{L}_{\mathrm{GM}}^{(m)} + \lambda_{\mathrm{align}} \mathcal{L}_{\mathrm{align}}^{(m)}. \tag{45}$$

For $k = 0, \ldots, K_{\mathrm{mem}} - 1$, client $m$ updates $\mathcal{M}_{\mathrm{syn},m}^{(r,k)}$ using this objective as described in Eq. (11). Only the resulting synthetic-graph update $\Delta_m^{(r)}$ is transmitted to the server.

Table 8: Partition statistics for three clients over the corpus. Local/Cross count questions whose annotated supporting passages are all held by the assigned client (Local) or not (Cross), reported separately for the 800 training, 200 validation and 1,000 test questions.

| Dataset | Setting | Passages/client | | | Train | | Val | | Test | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | Local | Cross | Local | Cross | Local | Cross |
| HotpotQA | Random | 6524 | 6527 | 6532 | 82 | 718 | 25 | 175 | 95 | 905 |
| | $\alpha = 1.0$ | 5287 | 6806 | 7490 | 105 | 695 | 34 | 166 | 146 | 854 |
| | $\alpha = 0.1$ | 5201 | 8508 | 5874 | 147 | 653 | 31 | 169 | 180 | 820 |
| 2WikiMultiHopQA | Random | 3782 | 3776 | 3777 | 70 | 730 | 14 | 186 | 94 | 906 |
| | $\alpha = 1.0$ | 3400 | 3361 | 4574 | 89 | 711 | 16 | 184 | 104 | 896 |
| | $\alpha = 0.1$ | 3574 | 3371 | 4390 | 100 | 700 | 28 | 172 | 148 | 852 |
| MuSiQue | Random | 8481 | 8843 | 8559 | 82 | 718 | 27 | 173 | 96 | 904 |
| | $\alpha = 1.0$ | 9930 | 9489 | 6464 | 81 | 719 | 15 | 185 | 91 | 909 |
| | $\alpha = 0.1$ | 5346 | 10717 | 9820 | 117 | 683 | 26 | 174 | 109 | 891 |

# C Detailed experimental settings

## C.1 Datasets and Federated Partitioning

**Dataset.** We evaluate on three multi-hop QA benchmarks: HotpotQA Yang et al. (2018), MuSiQue Trivedi et al. (2022), and 2WikiMultiHopQA Ho et al. (2020). **QA sets and retrieval corpus.** For each dataset, we use a training pool of 1,000 questions and a separate test set of 1,000 questions sampled from the benchmark validation split. We construct the retrieval corpus by pooling the candidate passages associated with both question sets, including supporting and distractor passages, before client allocation. The merged corpora contain 19,583 passages for HotpotQA, 25,883 for MuSiQue, and 11,335 for 2WikiMultiHopQA. Test-associated passages are available for graph construction and memory learning, while test questions and answers are excluded from training.

**Partitioning.** We consider random allocation and topic-skewed allocation with $\alpha \in \{1.0, 0.1\}$. Random allocation divides shuffled articles approximately equally among clients. For topic-skewed allocation, we embed passage titles and text using frozen `all-MiniLM-L6-v2`, average window embeddings within passages and passage embeddings within articles, and $\ell_2$-normalize the article representations. We then construct 20 K-means clusters per dataset, reusing these topic labels across both $\alpha$ settings. For each topic $k$, we sample $\boldsymbol{\pi}_k \sim \mathrm{Dirichlet}(\alpha \mathbf{1}_3)$ and allocate its articles using largest-remainder rounding. Smaller $\alpha$ favors stronger topic concentration; the allocation also introduces unequal client corpus sizes.

**Question ownership and support coverage.** Our main experiments use round-robin question ownership: question $i$ is assigned to client $i \bmod 3$ using its fixed zero-based position in the question list. This assignment is independent of the corpus partition, so topic skew is introduced only in the client corpora, while the query distribution remains approximately IID across clients. Consequently, this setting induces a high proportion of cross-client questions and serves as a stress test for shared retrieval under non-local evidence requirements.

A question is *local-answerable* if all of its annotated supporting articles are held by the assigned client, and *cross-client* if at least one supporting article is held by another client. The global question pool and round-robin ownership are fixed across partition settings. Table 8 summarizes the resulting corpus sizes and support availability for the main experiments.

**Topic-matched question ownership.** To test whether the results depend on the deliberately cross-client-heavy round-robin assignment, we additionally evaluate a topic-matched ownership scheme in which the query distribution follows the realized topic skew of the client corpora. Each question is assigned to the topic proxy of its annotated supporting articles, using the modal proxy when multiple topics occur. Questions of topic $k$ are then allocated across clients according to the realized corpus proportions $\pi_k$ for that topic using the largest-remainder rule. This ablation uses seed 42. The same

Table 9: Partition statistics under topic-matched question ownership. Local/Cross count questions whose annotated supporting passages are all held by the assigned client (Local) or not (Cross), reported separately for the 800 training, 200 validation, and 1,000 test questions.

| Dataset | Setting | Passages/client | | | Train | | Val | | Test | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | C1 | C2 | C3 | Local | Cross | Local | Cross | Local | Cross |
| HotpotQA | $\alpha = 1.0$ | 5287 | 6806 | 7490 | 180 | 620 | 47 | 153 | 234 | 766 |
| | $\alpha = 0.1$ | 5201 | 8508 | 5874 | 389 | 411 | 98 | 102 | 468 | 532 |
| 2WikiMultiHopQA | $\alpha = 1.0$ | 3400 | 3361 | 4574 | 133 | 667 | 34 | 166 | 159 | 841 |
| | $\alpha = 0.1$ | 3574 | 3371 | 4390 | 297 | 503 | 77 | 123 | 386 | 614 |
| MuSiQue | $\alpha = 1.0$ | 9930 | 9489 | 6464 | 141 | 659 | 20 | 180 | 173 | 827 |
| | $\alpha = 0.1$ | 5346 | 10717 | 9820 | 256 | 544 | 53 | 147 | 299 | 701 |

ownership rule is applied to the training, validation, and test questions, while the corpus partition is kept fixed within this ablation.

Topic matching substantially increases local support coverage as topic heterogeneity grows, while cross-client questions remain the majority in all evaluated settings. Table 9 reports the resulting support coverage, and Tables 13 and 14 report local- and cross-client F1 under moderate and strong topic heterogeneity, respectively.

### C.2 Models and Implementation Details

**Model.** We use Qwen2.5-7B as the LLM backbone, its weights remain frozen throughout training. Text embeddings and graph node features are produced by a all-MiniLM-L6-V2 SentenceTransformer which produces a 384-dimensional embedding. During training, these node representations are encoded by a four-layer graph convolutional network(GCN) with a hidden dimension of 384. Mean pooling over the resulting node embeddings produces a graph-level representation $\mathbf{g} \in \mathbb{R}^{384}$. A two-layer MLP projector with dimensions $384 \to 2048 \to 3584$ and a GELU activation then maps $\mathbf{g}$ into the LLM embedding space, yielding a soft-prompt token in $\mathbb{R}^{3584}$.

**Comparison protocol and reader adaptation.** All baselines use the same LLM backbone as SynFedRAG, the same QA training split, and the same answer-generation prompt. Baselines without a dedicated reader-training stage use a RoLoRA-adapted checkpoint trained on this split, without further baseline-specific reader fine-tuning.

**Graph construction and retrieval.** Each client constructs its private entity–sentence–passage graph following LinearRAG. For each question, we retrieve the top 5 local passages and construct their passage–entity evidence subgraph as described in Appendix B.4.1. The retrieved passage text is truncated to $L = 1024$ tokens. Synthetic retrieval considers all $K_g$ nodes, with question-conditioned weights computed using temperature $\tau = 0.1$ (Appendix B.4.2). The PGE-induced soft adjacency is used consistently during training and inference.

**Optimization and hyperparameters.** We use Adam for all optimization stages. For graph condensation, we set $\lambda_{\text{mmr}} = 0.3$, $b_S = b_P = 3$, and weight the IDF and PageRank terms by $\lambda_{\text{idf}} = 1.0$ and $\lambda_{\text{pr}} = 0.5$, respectively; PageRank uses 20 iterations with damping $0.85$. Neighborhood condensation uses hop budgets $(1, 3, 2)$ with weights $(0.4, 0.4, 0.2)$, chunk budget $b_c = 8$, fusion coefficient $\rho = 0.5$, and $k_{\text{topo}} = 8$. The unscaled neighborhood and chunk softmaxes correspond to $\tau_n = \tau_c = 1$. For the condensation objective, we use $\lambda_{\text{rep}} = 1.0$, $\lambda_{\text{div}} = 0.1$, $\tau_{\text{ret}} = \tau_a = 0.1$, $\tau_\pi = \sqrt{128}$, and $\delta_t = 0.5$. Condensed features are refined for 1,000 steps with learning rate $10^{-2}$.

For the synthetic graph, we use $K_g = 128$ nodes and a PGE with hidden dimension 256, top-8 connectivity, and 16-dimensional type embeddings. We set $\lambda_{\text{reg}} = 0.1$, initialize the synthetic graph for 200 steps with learning rate $10^{-2}$, and use $\eta_{\text{reg}} = 10^{-2}$ for the server update. During client-side refinement, we set $\lambda_{\text{gm}} = \lambda_{\text{align}} = 0.1$ and perform $K_{\text{mem}} = 200$ updates with $\eta_{\text{syn}} = 10^{-3}$. The graph-guided prompt module is optimized with learning rate $10^{-4}$ and QA batch size 128.

**Federated setup.** We use $M = 3$ clients and $T = 10$ communication rounds with full client participation. Each client performs three epochs of local prompt-module training per round before graph refinement. Experiments run on a single NVIDIA GeForce RTX 4090 GPU with 24 GB of memory.

**Statistical reporting.** Unless otherwise stated, we report mean and standard deviation over three runs with seeds 42, 43, and 44. Each run uses a single seed for both corpus partitioning and stochastic training operations. We report exact match and token-level F1 for overall QA performance, and F1 separately for local-answerable and cross-client questions.

# D Evaluation results

## D.1 Communication and Computational Efficiency

We evaluate the communication cost and per-round training time of SynFedRAG under different synthetic-graph budgets. We distinguish total offline communication from retrieval-knowledge communication, which excludes prompt-module exchanges but includes the parameters used to represent the synthetic graph. After the final prompt module and synthetic graph are deployed locally, inference requires no communication with the server or other clients.

**Communication cost.** We estimate offline communication during initialization and federated training, including the one-time upload of client-condensed graphs, bidirectional exchanges of prompt-module parameters, synthetic-graph downloads, and graph-update uploads:

$$\begin{aligned} C_{\text{total}} &= C_{\text{init}} + C_{\text{model}} + C_{\text{graph}}, \\ C_{\text{init}} &= \sum_{m=1}^{M} \text{Bytes}(\widetilde{G}_m), \\ C_{\text{model}} &= \sum_{r=0}^{T-1} \sum_{m \in S_r} \left( C_{W,\downarrow}^{(r,m)} + C_{W,\uparrow}^{(r,m)} \right), \\ C_{\text{graph}} &= \sum_{r=0}^{T-1} \sum_{m \in S_r} \left( C_{\text{syn},\downarrow}^{(r,m)} + C_{\Delta\text{syn},\uparrow}^{(r,m)} \right). \end{aligned} \tag{46}$$

Here, $C_{\text{init}}$ denotes the one-time upload of client-condensed graphs, and $C_{\text{model}}$ accounts for exchanges of the complete GNN encoder and projector. For graph communication, each participating client downloads $X_{\text{syn}}^{(r)}$ and $\theta_{\text{PGE}}^{(r)}$, then uploads $\Delta_m^{(r)}$, containing the corresponding updates after local refinement. Both communication directions are counted across participating clients. The final deployment broadcast after the last round is excluded. Communication volumes are reported in MiB.

**Effect of synthetic-graph capacity.** We vary the synthetic-node budget $K_g \in \{32, 64, 128, 256\}$ to examine its effect on cross-client QA performance and system cost. Cross-client F1 is evaluated on questions whose annotated supporting passages are not fully available in the querying client's corpus. Table 10 reports Cross F1, per-round training time, synthetic-graph state size, and estimated total offline communication.

Among the evaluated budgets, $K_g = 128$ yields the highest reported cross-client F1 on all three benchmarks. Increasing the budget from 32 to 128 improves performance, particularly on MuSiQue and 2WikiMultiHopQA, although the gains are not monotonic. Further increasing $K_g$ to 256 provides no additional improvement. Meanwhile, the synthetic-graph state grows from 0.05 to 0.63 MiB, and total offline communication increases from 693.0 to 716.0 MiB, with little variation in per-round training time. These results support $K_g = 128$ as the default budget in our evaluated setting, while showing that larger synthetic graphs do not necessarily improve QA performance.

**Capacity and convergence analysis.** Figure 2 shows overall QA F1 across communication rounds for different synthetic-node budgets. Performance generally improves during refinement. Because

Table 10: Effect of the synthetic-node budget $K_g$ on cross-client QA performance and system cost.

| | HotpotQA | | MuSiQue | | 2Wiki | | Communication | |
|---|---|---|---|---|---|---|---|---|
| $K_g$ | Cross F1 | Time (min) | Cross F1 | Time (min) | Cross F1 | Time (min) | Mem. (MiB) | Total (MiB) |
| 32 | 43.39 | 43.7 | 31.92 | 42.6 | 35.09 | 40.6 | 0.05 | 693.0 |
| 64 | 42.28 | 43.5 | 34.18 | 42.6 | 38.21 | 40.6 | 0.11 | 695.3 |
| 128 | **44.70** | 45.8 | **34.53** | 42.6 | **41.08** | 40.6 | 0.25 | 701.0 |
| 256 | 44.58 | 43.9 | 34.17 | 42.8 | 39.98 | 40.7 | 0.63 | 716.0 |

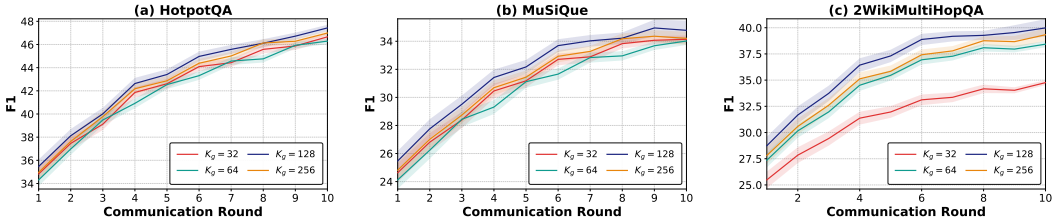



Figure 2: **Effect of synthetic graph capacity throughout federated optimization.** Overall QA F1 over communication rounds for synthetic-node budgets $K_g \in \{32, 64, 128, 256\}$ on (a) HotpotQA, (b) MuSiQue, and (c) 2WikiMultiHopQA. Solid lines show the mean across runs, and shaded regions show standard deviation.

both the graph-guided prompt module and the synthetic graph are updated, these trajectories reflect their combined optimization. Together with Table 10, the figure characterizes performance sensitivity around the default budget of $K_g = 128$.

**Communication protocol comparison.** We further compare the retrieval-knowledge communication of SynFedRAG, DGRAG, and FD-RAG. Unlike the total offline communication in Eq. (46), this comparison excludes model-adaptation exchanges for all methods. For SynFedRAG, PGE parameters remain part of the retrieval-knowledge payload.

SynFedRAG uploads client-condensed graphs during initialization and subsequently synchronizes the shared synthetic graph during federated refinement. DGRAG exchanges graph summaries and additionally incurs query-time communication when retrieving cross-client evidence. FD-RAG constructs a shared retrieval memory that can be used locally at inference time. For SynFedRAG, the retrieval-knowledge communication is

$$C_{\text{knowledge}} = C_{\text{init}} + C_{\text{graph}}. \tag{47}$$

The server sends the synthetic-graph state to each participating client, which returns its local update $\Delta_m^{(r)}$. The resulting graph communication is

$$C_{\text{graph}} = \sum_{r=0}^{T-1} \sum_{m \in S_r} \left[ \text{Bytes}\left(X_{\text{syn}}^{(r)}, \theta_{\text{PGE}}^{(r)}\right) + \text{Bytes}\left(\Delta_m^{(r)}\right) \right]. \tag{48}$$

If updates are transmitted as dense tensors with the same shapes and numerical precision as the downloaded state, both directions have equal payload sizes. Under full participation and a fixed payload size, this reduces to $C_{\text{graph}} = 2MT\,\text{Bytes}(X_{\text{syn}}, \theta_{\text{PGE}})$.

Table 11 separates initialization, synchronization, and query-time communication. For SynFedRAG, the reported estimates comprise 25.22 MiB for initialization and 1.50 MiB per round for aggregate graph downloads and update uploads across three clients. Over ten rounds, these sum to $25.22 + 10 \times 1.50 = 40.22$ MiB. FD-RAG's one-time memory upload and distribution are estimated at 75.13 MiB. After deployment, both methods use locally available retrieval memory without additional query-time communication in the evaluated configuration.

Table 11: Estimated retrieval-knowledge communication with three clients on HotpotQA. Upload and download traffic are counted across clients. Model-adaptation exchanges are excluded.

| Method | Initialization (MiB) | Synchronization (MiB/round) | Online (MiB/event) | Total (MiB) |
|---|---|---|---|---|
| SynFedRAG | 25.22 | 1.50 | 0 | 40.22 |
| DGRAG | 20.40 | 0 | 0.047 | 62.37 |
| FD-RAG | 75.13 | 0 | 0 | 75.13 |

For DGRAG, the estimated total communication is

$$C_{\mathrm{DGRAG}} = C_{\mathrm{init}}^{\mathrm{DGRAG}} + N_{\mathrm{cross}} C_{\mathrm{remote}}, \tag{49}$$

where $N_{\mathrm{cross}}$ counts remote-retrieval events under the assumed workload, rather than necessarily equaling the number of questions with cross-client gold support. With 893 events, the estimated DGRAG communication is $20.40 + 893 \times 0.047 = 62.37$ MiB.

Communication depends on both the offline exchange protocol and the query-time workload. SynFedRAG exchanges retrieval knowledge during initialization and federated refinement, whereas FD-RAG constructs and distributes a shared retrieval memory. After deployment, both methods support local inference without additional query-time communication in the evaluated configuration.

### D.2 Comparison with Federated LoRA Baselines

**Adaptation mechanisms.** We compare SynFedRAG with FLoRA, FlexLoRA, and RoLoRA using Qwen2.5-7B. The LoRA baselines adapt query and value projections with ranks $r \in \{8, 16, 32, 64\}$, whereas SynFedRAG trains an external graph-guided prompt module and a shared synthetic graph. Both approaches keep the pretrained backbone weights frozen. SynFedRAG uses its default configuration throughout the comparison.

**Communication accounting.** Table 12 reports trainable parameter counts, upload and download costs per round, total offline communication, and HotpotQA F1. Under the evaluated configurations, the three LoRA baselines share the same parameter counts and communication estimates at each rank. For SynFedRAG, total offline communication includes client-condensed graph uploads, prompt-module exchanges, and synthetic-graph synchronization. This total differs from retrieval-knowledge communication, which excludes prompt-module exchanges.

**Performance–communication trade-off.** Increasing the LoRA rank improves the reported mean F1 for all three baselines, while communication grows proportionally with rank. At $r = 64$, RoLoRA achieves the highest baseline F1 of 25.19, compared with 47.71 for SynFedRAG, a difference of 22.52 points. Under the reported communication estimates, SynFedRAG requires more total communication than the rank-8 and rank-16 configurations, but less than the rank-32 and rank-64 configurations. Thus, SynFedRAG achieves higher F1 across the evaluated configurations, while its relative communication cost depends on the LoRA rank.

### D.3 Topic-Matched Question Ownership

Our main experiments use round-robin question ownership, which is independent of the topic-skewed corpus partition and therefore creates a cross-client-heavy retrieval setting. To test whether the results depend on this choice, we additionally evaluate a topic-matched ownership scheme in which the query distribution follows the realized topic distribution of each client corpus.

Each question is assigned to the topic proxy of its annotated supporting articles, using the modal proxy when multiple topics occur. Questions of topic $k$ are then allocated across clients according to the realized corpus proportions $\pi_k$, using the same seed and largest-remainder rule as the corpus partition. The same rule is applied to training, validation, and test questions, while the corpus partition is unchanged. Table 9 reports the resulting support coverage. Topic matching increases the fraction of locally answerable questions, while cross-client questions remain the majority across all evaluated settings. Tables 13 and 14 report local- and cross-client F1.

Table 12: HotpotQA performance and communication cost at different LoRA ranks. For each LoRA row, parameter counts and communication costs are shared by FLoRA, FlexLoRA, and RoLoRA under the evaluated configurations, while F1 is reported separately. LoRA adapts only query and value projections. SynFedRAG uses its default configuration. F1 is evaluated under topic-skewed partitions with $\alpha = 1.0$. Dashes indicate results not yet available.

| | | Communication | | | HotpotQA F1 | | |
|---|---|---|---|---|---|---|---|
| **Configuration** | **Trainable params. (M)** | **Upload (MiB/round)** | **Download (MiB/round)** | **Total (MiB)** | **FLoRA** | **FlexLoRA** | **RoLoRA** |
| LoRA, $r = 8$ | 2.52 | 14.44 | 4.81 | 192.5 | $13.06 \pm 1.43$ | $12.78 \pm 0.61$ | $12.67 \pm 0.44$ |
| LoRA, $r = 16$ | 5.05 | 28.88 | 9.62 | 385.0 | $18.51 \pm 1.14$ | $18.22 \pm 0.95$ | $18.43 \pm 1.04$ |
| LoRA, $r = 32$ | 10.09 | 57.75 | 19.25 | 770.0 | $20.17 \pm 0.84$ | $21.26 \pm 1.07$ | $22.86 \pm 0.91$ |
| LoRA, $r = 64$ | 20.19 | 115.50 | 38.50 | 1540.0 | $23.26 \pm 0.57$ | $24.41 \pm 0.43$ | $25.19 \pm 1.22$ |
| SynFedRAG | 8.73 | 50.68 | 16.89 | 701.0 | $\mathbf{47.71} \pm 1.03$ | | |

Table 13: Local- and cross-client F1 under topic-matched question ownership with $\alpha = 1.0$.

| | | HotpotQA | | MuSiQue | | 2WikiMultiHopQA | |
|---|---|---|---|---|---|---|---|
| **Group** | **Method** | **Local** | **Cross** | **Local** | **Cross** | **Local** | **Cross** |
| FL + Local GraphRAG | G-retriever | 50.58 | 35.14 | 25.04 | 18.94 | 39.08 | 30.72 |
| | GRAG | 52.33 | 34.12 | 24.78 | 21.64 | 44.82 | 33.19 |
| FL + Local RAG | FLARE | 52.64 | 25.49 | 22.43 | 11.67 | 44.25 | 30.18 |
| | Dense RAG | 51.81 | 9.64 | 20.44 | 11.30 | 33.64 | 29.80 |
| Federated RAG | DGRAG | 48.04 | 39.35 | 21.83 | 32.97 | 43.56 | 37.13 |
| | FD-RAG | 56.87 | 39.02 | 35.24 | 33.16 | 41.96 | 37.93 |
| | **SynFedRAG (Ours)** | 60.58 | 40.65 | 38.25 | 34.61 | 40.25 | 39.04 |

### D.4 Privacy Leakage Analysis

SynFedRAG keeps raw documents local, but its shared representations may still reveal information about private corpora. We distinguish the information exposed to the server from that exposed to participating clients, and evaluate membership leakage from the corresponding graph representations.

**Adversaries and exposed information.** An honest-but-curious server receives each client's condensed graph during initialization, including passage-node features derived from real private passages. In each subsequent round, it receives the prompt module parameters $W_m^{(r)}$ and synthetic-graph updates $\Delta_m^{(r)}$ uploaded by participating clients. It also observes the global prompt module and synthetic graph that it maintains. Condensed graphs are transmitted once, but the server may retain them and subsequent updates.

An honest-but-curious client $a$ observes its own private corpus and local training states, together with the global prompt module $W^{(r)}$ and synthetic graph $\mathcal{M}_{\text{syn}}^{(r)}$ received during the protocol. It may retain all received versions and use them to infer information about other clients. Because the shared graph is optimized using contributions from multiple clients, its distribution creates persistent cross-client exposure to representations learned from their private corpora. We therefore evaluate this exposure separately.

**Membership leakage from individual states.** The first evaluation asks whether a candidate passage belongs to a specified client's corpus. For each target client, we construct a balanced set of member passages from its corpus and non-member passages from the same benchmark. Using the fixed text encoder employed by SynFedRAG, we compute the maximum and top-$k$ mean cosine similarities between each candidate passage and the exposed node features. A lightweight classifier predicts membership from these statistics and is evaluated on a disjoint attack-test split. Table 6 reports AUC and TPR at 1% FPR. The full private graph serves only as a reference exposure condition and is not transmitted by SynFedRAG.

Table 14: Local- and cross-client F1 under topic-matched question ownership with $\alpha = 0.1$.

| | | HotpotQA | | MuSiQue | | 2WikiMultiHopQA | |
|---|---|---|---|---|---|---|---|
| **Group** | **Method** | **Local** | **Cross** | **Local** | **Cross** | **Local** | **Cross** |
| FL + Local GraphRAG | G-retriever | 44.52 | 35.91 | 27.43 | 20.74 | 38.19 | 36.02 |
| | GRAG | 48.39 | 33.76 | 23.61 | 20.35 | 40.22 | 36.98 |
| FL + Local RAG | FLARE | 52.96 | 24.87 | 21.63 | 12.34 | 43.11 | 29.67 |
| | Dense RAG | 40.49 | 13.07 | 20.84 | 10.62 | 37.63 | 27.79 |
| Federated RAG | DGRAG | 48.22 | 42.82 | 21.07 | 32.34 | 43.24 | 37.09 |
| | FD-RAG | 56.12 | 43.17 | **35.91** | 33.29 | 41.37 | 37.54 |
| | **SynFedRAG (Ours)** | **64.27** | **52.40** | 35.74 | **35.87** | **43.78** | **45.26** |

The full private graph yields nearly perfect membership inference across all three datasets. Condensation reduces attack performance, but the condensed graphs still yield 67.1–72.9% AUC and 20.6–25.9% TPR, indicating that their upload exposes substantial membership information to the server under this attack. Synthetic graph yields AUC close to 50% both initially and after refinement, with no consistent increase across datasets. This suggests that direct passage-to-node similarity is less informative for the evaluated client-specific membership task in synthetic graph. However, its TPR remains above the 1% random baseline.

**Client-side inference across rounds.** To evaluate persistent cross-client exposure, we consider each client $a$ in turn as the attacker. Here, the target is membership in the union of the other clients' corpora, rather than attribution to a particular client. Positive candidates are passages from these other corpora, excluding passages also present in the attacker's own corpus. Negative candidates are passages from the same benchmark that are absent from every participating client's corpus and were not used to construct or optimize the shared graph.

We compare two observation conditions. The *final-snapshot* attack uses similarity statistics from the last synthetic graph received by the attacker. The *accumulated-snapshots* attack concatenates the same statistics from every received graph version in chronological order. Both conditions use identical candidate passages, attack-data splits, and classifier-selection procedures.

**Scope.** These attacks evaluate graph-representation leakage under specified membership tasks. They do not exhaust the adversaries' available information: attacks jointly exploiting prompt module parameters, client updates, or private local training data remain outside this evaluation. Moreover, low leakage from synthetic graph does not undo the server's earlier exposure to condensed graphs. Our findings therefore provide an empirical assessment under the evaluated attacks, not a formal privacy guarantee.

**Readout-aware membership inference.** The preceding attack uses similarity statistics computed directly from the synthetic-node features $\mathbf{X}_{\text{syn}}$. We additionally consider a stronger white-box attacker that uses the complete trained synthetic-graph readout defined in Appendix B.3.2, including the GNN-contextualized node representations, query-conditioned weights, and projected synthetic context. The readout-aware classifier augments the original raw-feature statistics with statistics derived from $\mathbf{H}_{\text{syn}}$, $\alpha(p)$, and $\mathbf{z}_{\text{syn}}(p)$. We keep the candidate sets, attack split, classifier protocol, and attack seed identical to the raw-feature attack. Table 15 reports target-client membership inference results for the final synthetic graph under raw and readout-aware attacks.

The readout-aware attack does not consistently improve client-specific membership inference over the same-seed raw-feature baseline: its AUC is $47.0$, $47.2$, and $53.3$, compared with $48.6$, $50.3$, and $53.6$ for the raw attack. TPR at 1% FPR likewise remains low without a consistent increase across datasets. As robustness checks, using all $K_g = 128$ nodes for the raw attack gives AUCs of $47.7$, $48.8$, and $51.2$, while using $\mathbf{X}_{\text{syn}}$, $\theta_{\text{PGE}}$, $\theta_{\text{GNN}}$, and $\theta_{\text{proj}}$ from the same communication round gives readout-aware AUCs of $48.5$, $47.3$, and $53.3$. Thus, the near-chance membership results are not simply an artifact of attacking only the raw synthetic-node features. This remains an empirical observation under the evaluated attacks rather than a formal privacy guarantee.

Table 15: Target-client passage membership inference from the final synthetic graph. Raw and readout-aware attacks use identical candidate sets, attack splits, classifier protocol, and attack seed. AUC and TPR at 1% FPR are reported in percentages.

| | HotpotQA | | MuSiQue | | 2Wiki | |
|---|---|---|---|---|---|---|
| Attack | AUC | TPR | AUC | TPR | AUC | TPR |
| Raw $\mathbf{X}_{\text{syn}}$ | 48.6 | 1.9 | 50.3 | 1.7 | 53.6 | 1.4 |
| Readout-aware | 47.0 | 4.2 | 47.2 | 1.1 | 53.3 | 2.5 |
| Random guess | 50.0 | 1.0 | 50.0 | 1.0 | 50.0 | 1.0 |

### D.5 Effect of Regularization on Node Diversity

The synthetic nodes are initialized with distinct representations, providing multiple candidate directions for query-conditioned readout. However, graph propagation and subsequent optimization can make different nodes increasingly similar, reducing the effective diversity of the synthetic memory. We therefore examine whether $\mathcal{L}_{\text{reg}}$ helps preserve these distinctions throughout training. Figure 3 compares the evolution of synthetic-node representations using t-SNE with and without $\mathcal{L}_{\text{reg}}$ on HotpotQA, while keeping the client-side condensation objectives unchanged. Without regularization, several synthetic nodes progressively collapse into compact groups after Phase I and Phase II, whereas these groups are much less pronounced when $\mathcal{L}_{\text{reg}}$ is applied. This behavior is consistent with neighborhood-induced homogenization in message-passing GNNs (Nguyen et al., 2023), where nodes aggregating similar neighborhood information become less distinguishable after propagation. Since graph-level alignment constrains only the pooled representation, it does not directly prevent such node-level redundancy. The regularizer therefore helps maintain distinctions among synthetic-node representations that can be exploited by query-conditioned aggregation, rather than promoting geometric separation as an objective by itself.

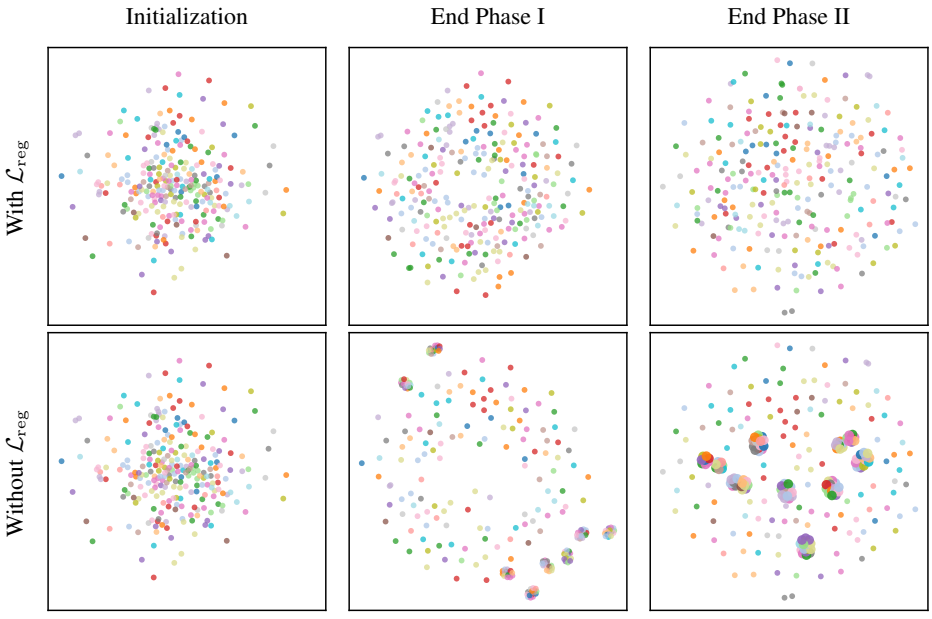



Figure 3: Effect of regularization on synthetic-node representations. We visualize $H_{\text{syn}}$ using t-SNE; quantitative diversity metrics are computed in the original embedding space. Colors denote each synthetic node's nearest topic cluster, assigned by cosine similarity to its initial embedding and fixed across panels.

### D.6 Qualitative Analysis

To complement the aggregate results, we examine representative HotpotQA examples in greater detail. Table 16 compares the predictions of HippoRAG, FLARE, and SynFedRAG, illustrating cases where cross-client evidence or answer-form variation affects the final prediction. Table 17 further analyzes two of these examples by reporting the highest-weighted synthetic nodes and their query-conditioned readout weights, providing a closer view of which parts of the shared synthetic graph are emphasized for each question.

Table 16: Qualitative examples on HotpotQA. We compare HippoRAG, FLARE, and SynFedRAG. Red indicates an incorrect answer.

| | HippoRAG | FLARE | SynFedRAG (Ours) |
|---|---|---|---|
| **Question** | What is the city of the headquarters of the American research and scientific development company where scientist Ravi Sethi worked as a computer scientist? **Gold answer:** `Murray Hill, New Jersey` | | |
| **Output** | "Insufficient information to determine the headquarters location from the provided context." | "The headquarters of the American research and scientific development company is not specified in the given context." | "Murray Hill, New Jersey." |
| **Explanation** | The passage containing the gold answer is present at the querying client but is missed by local retrieval. The synthetic nodes associated with Ravi Sethi, Bell Labs, and Avaya Labs Research are listed in Table 17; all three have the gold-bearing "Ravi Sethi" passage as their nearest passage. SynFedRAG answers correctly while both baselines decline to answer, suggesting that synthetic memory may help when local retrieval misses available evidence. | | |
| **Question** | If You Ever Get Lonely was covered by what Lyric Street Records-affiliated band? **Gold answer:** `Love and Theft` | | |
| **Output** | "The question cannot be definitively answered with the provided information." | "Jackson" | "Love and Theft." |
| **Explanation** | The querying client lacks the gold answer in its local text. The frozen backbone answers "Bowling for Soup," and the local-only variant answers "Papa Roach." SynFedRAG answers correctly in the individually reproduced run. Table 17 reports the highest-weighted synthetic nodes and their nearest passages at the other client. | | |
| **Question** | What was the name of the lawyer who took the main role in the report that helped land Pete Rose a lifetime suspension from baseball? **Gold answer:** `John M. Dowd` | | |
| **Output** | "John M. Dowd." | "The name of the lawyer who took the main role in the report that helped land Pete Rose a lifetime suspension from baseball is John M. Dowd." | "John M. Dowd." |
| **Explanation** | All three methods answer correctly on this local-answerable question. | | |
| **Question** | Which Ottoman Sultan was Gevherhan Sultan's grandfather? **Gold answer:** `Suleiman I` | | |
| **Output** | "Suleiman the Magnificent." | "Suleiman the Magnificent." | "Suleiman the Magnificent." |
| **Explanation** | All three outputs give a valid name for Suleiman I. They fail exact match but receive partial token-F1 credit because "Suleiman" matches the reference answer. This is an alias-related evaluation artifact. | | |

**Synthetic-node analysis.** Table 17 shows different readout patterns in the two examples. For the cross-client question, the reported nodes refer to Eric Gunderson, Lyric Street Records, and Love and Theft; their weights sum to 0.24, so most of the synthetic readout remains distributed over other

Table 17: Synthetic nodes for two examples in Table 16.

| **Question** | **Node** | **Node entity** | $\alpha_i(q)$ |
|---|---|---|---|
| "If You Ever Get Lonely" | 88 | Eric Gunderson | 0.14 |
| | 41 | Lyric Street Records | 0.06 |
| | 12 | Love and Theft | 0.04 |
| Ravi Sethi (headquarters) | 1 | Ravi Sethi | 0.30 |
| | 34 | Bell Labs | 0.22 |
| | 82 | Avaya Labs Research | 0.15 |

nodes. Their nearest passage at client 1 is "Love and Theft (duo)." For the Ravi Sethi question, the three reported nodes receive approximately 0.67 of the total weight and are nearest to the gold-bearing passage at the querying client, which local retrieval misses. These observations show that the question-conditioned readout selects relevant nodes in both settings, but node weights and nearest passages alone do not establish which information caused the LLM's answer.