Policy-Based Access Control in Federated Clinical Question Answering

by

Alice Chen

S.B. in Computer Science and Engineering
Massachusetts Institute of Technology (2024)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2024

Authored by: Alice Chen
Department of Electrical Engineering and Computer Science
May 10, 2024

Certified by: Lalana Kagal
Principal Research Scientist, MIT CSAIL
Thesis Supervisor

Accepted by: Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Policy-Based Access Control in Federated Clinical Question Answering

by

Alice Chen

Submitted to the Department of Electrical Engineering and Computer Science
on May 10, 2024, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Retrieval augmented generation (RAG) has recently expanded large language model versatility in answering domain-specific questions using dynamic external knowledge bases, particularly demonstrating promise in assisting clinical settings. However, due to its sensitive nature, patient medical data often requires retrieval to be federated across a decentralized network of hospital institutions, each maintaining internal databases and access control policies. Applying standard RAG to clinical question-answering tasks is complicated by the lack of an interface for hospital resource owners to regulate and restrict access to sensitive clinical documents during retrieval, which is essential for model feasibility in practice. We propose to leverage federated RAG retrieval for clinical trends inference across distributed medical records while adding authorization security mechanisms during retrieval to guarantee security of patient data. We propose (i) user identity authentication administered through a trusted federation of per-hospital OpenID Connect servers, (ii) a framework for integrating policy-based access control (PBAC) security mechanisms at flexible granularity into a federated RAG system to restrict medical data access based on user role attributes, and (iii) ClinicalTrendQA, a novel dataset to evaluate model performance for synthesizing clinical trends grounded on decentralized patient EHR information. To facilitate evaluation of our authorization PBAC framework on protecting information leakage during retrieval, we additionally present a federated 3-hospital case study and demonstrate that the same ClinicalTrendQA query under different user profiles holding varying degrees of access privileges observes the expected EHR information reduction. We also analyze metrics concerning the impact of this retrieval loss on end-to-end response quality against federated insecure and centralized RAG baselines.

Thesis Supervisor: Lalana Kagal
Title: Principal Research Scientist, MIT CSAIL

Acknowledgments

I would like to express my gratitude to my research supervisor, Lalana Kagal, for her continued guidance throughout this project, and Emily Jiang, another M.Eng student with whom I worked on the federated RAG system in close collaboration. I also appreciate the support of the Decentralized Information Group (DIG); in particular, Irene Tenison and Zachary Johnson for their help in deploying various LLM models and Minseok Jung for his advice on the HELM evaluation dataset.

Finally, I would like to thank my mother, Jingru Wang, and father, Yongming Chen, for their unconditional support and encouragement throughout my life, as well as all of the friends I had by my side these four years.

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 1

Introduction and Motivation

Large-language models (LLMs) have demonstrated impressive success in extracting information from unstructured text, with diverse applications for answering free-form questions, generating fluent and natural conversational dialogue, and translating text between languages among other language inference tasks [4] [9]. However, practical application of LLMs suffer from several inherent flaws:

1. **LLM models are trained on a static dataset** of examples and thus they struggle with maintaining **recency** of information. In particular, they can only generate responses from the fixed knowledge stored in their parameters at the last time they were retrained. This means that LLMs are slow to react to updated information, since incorporating new knowledge requires the computational resource-costly task of fine-tuning the LLM parameters on every new dataset update. For example, a general question-answering LLM trained on events up to 2019 would not be aware of the 2020 U.S. presidential election.

2. **LLM models are prone to hallucinations**, which are erroneous but plausibly constructed responses either directly misrepresenting input document facts (intrinsic) or fabricating information absent from the LLM's trained input dataset (extrinsic) [30] [24]. On questions for which an LLM lacks sufficient knowledge, the preferred behavior is to decline to answer citing lack of information rather than invent "hallucinatory" outputs. Reliably ensuring the fidelity of

LLM question-answer output is particularly essential if making human decisions, such as basing clinical diagnoses on LLM responses.

3. **LLM models cannot flexibly handle NLP tasks for domain-specific knowledge** since it requires retraining the LLM model on the domain-specific datasets. This is very computationally expensive (i.e. OpenAI's GPT-3 contains 175 billion parameters [4]) and sometimes infeasible if the dataset contains sensitive data that cannot be used to tune hosted language models due to security concerns.

4. **LLM models lack interpretability of their responses**. It is not intuitive to understand which sources the LLM used to generate its responses or how the LLM weighted and synthesized information between the sources, which further complicates detecting incorrect hallucinations or biased output.

Retrieval-Augmented Generation (RAG) [22] is a promising framework that addresses these drawbacks by introducing non-parametric memory in an initial retrieval step in addition to the parametric LLM memory. The encoder-decoder retriever encodes the query and returns the top k text documents by similarity score to the query from a dense vector index pretrained on the external knowledge source. The retrieved documents are augmented to the original query and collectively passed to the original LLM to be cited as context during response generation, which also suggests potential for various interpretability techniques. Thus, the RAG paradigm allows for updating the knowledge base via fine-tuning of the dense vector index retriever without requiring retraining the generator LLM.

Clinical-Domain NLP Tasks

Patient clinical information is often organized into Electronic Health Records (EHR), which are composed of *structured* data, such as vital measurements and active diagnoses, as well as an abundant quantity of *unstructured* text data from doctor-written patient notes and discharge summaries [15]. Despite their utility in assisting patient clinical care decisions, EHR records are costly to review and maintain: nonsurgical physicians (from a size 155,000 sample) spend approximately 3.3 hours per day performing EHR tasks, with the bulk of the time across all specialties spent on chart review (33%), followed by documentation (24%) [14], which diverts valuable clinician time away from patients and is susceptible to human errors like missing EHR note details. Applying natural-language models to assist physicians during EHR review tasks has significant potential in allowing medical providers to devote more time to patient needs during encounters.

To capitalize on inference on these unstructured EHR notes, there has been significant interest in applying RAG or clinical-domain fine-tuned LLMs to assist medical professionals in making informed diagnostic decisions or other healthcare advice [1] [37] [31] [5]. Prior work, such as the closed-source, state-of-the-art Med-PaLM [31] and updated Med-PaLM-2 clinical LLMs [32], has successfully fine-tuned medical QA LLMs on non-patient-specific medical-school knowledge to pass US Medical Licensing Exam-style questions and other textbook clinical recall tasks. On the same standard medical benchmarks (MMLU-Medical, PubMedQA, MedMCQA, MedQA), the open-source Meditron LLM (available with 7B and 70B parameters) has also recently demonstrated performance surpassing Med-PaLM and within 10% of Med-PaLM-2 [5]. Unfortunately, both the medical QA task benchmarks and the performance of state-of-the-art clinical domain LLMs are predominantly focused on question-answering over *generalized* medical knowledge and thus only require fine-tuning over public clinical information without the challenges inherent to privacy-sensitive *personal* patient data.

Directly applying LLMs in patient-focused medical settings is complicated by the **distributed**, **dynamic**, and **privacy-sensitive** nature of personal healthcare data. Clinical data is decentralized across multiple hospital organizations and likely several internal departments (i.e. ICU, internal medicine, psychiatry) within each organization rather than aggregated in a central repository; furthermore, building such a centralized store of inter-hospital non-anonymized EHR data undesirably exposes patient data to third-party organizations or users without their explicit consent. Unlike factual medical knowledge, patient EHR records are liable to change on a frequent basis from updating health conditions. Most importantly, patient medical records are one of the most *sensitive* genres of information, for which guaranteeing privacy-protection is indispensable. Hospital organizations are very concerned with properly regulating EHR record access across healthcare workers performing different roles (i.e. doctors, nurses, administrative staff, police) [3]: for example, doctors may need complete EHR access to deliver proper medical care, while administrative staff should only be authorized to view less-sensitive information like dates of hospital admission. Likewise, the Health Insurance Portability and Accountability Act (HIPAA) prohibits feeding patient medical data into externally-hosted LLMs like ChatGPT [1].

Much existing clinical LLM research focuses on factual questions [31] answerable without personal health data or on single-patient diagnosis involving only 1 patient's EHR [37] [1], such as "What is the treatment for Anterior Cruciate Ligament (ACL) injuries in children?" or "Read the following patient's clinical note: `[NOTE]`. Answer step by step: is the patient at risk for gout, and why?", respectively. For the latter question task, the LLM model's [1] patient-specific clinical knowledge is isolated to the single provided note and thus the generated response is ignorant of any other medical history for that patient or their community. We propose that augmenting clinical LLMs with a prior retrieval step over patient EHR data will encourage more personalized and helpful responses for the specific patient(s) involved. In particular, we are interested in exploring the adjacent natural-language QA task of analyzing **clinical *trends* across patient EHR data** such as "*Across female patients at this practice* admitted with major depression and treated with Celexa, what have been common responses to the medication?". We believe that leveraging RAG-LLM models for clinical trend inference on specific patient data has potential to provide medical caregivers more localized and helpful responses.

To do so, we must adapt the standard open-domain RAG question-answering pipeline along 2 dimensions:

1. **Federation**. Unlike generalized clinical-domain knowledge, which requires only a static retrieval vector store from medical textbooks [35] or internet articles [37], patient EHR data is decentralized across multiple hospitals, so the standard

1 retriever RAG model must be expanded to federate retrieval across one or multiple hospital knowledge bases.

2. **Data Security**. Standard RAG systems expose the entire knowledge base during retrieval regardless of user identity, which is unacceptable for privacy-sensitive medical data. Hospitals must be able to protect and selectively control access to only authorized EHR records, such as via policy/attribute-based access control (PBAC). The interface should ideally support heterogeneous EHR-access security models at any granularity; for example, the pediatric and adult health departments within a hospital may require different health record access laws (i.e. parental guardian access rights). **This, assuming a federated RAG system, is the focus for our work.**

In addition to our primary contribution of enhancing federated RAG with authentication and authorization mechanisms (see Section 3), our secondary contribution includes a novel dataset to evaluate the model's performance on **clinical trend QA inference tasks** (see Section 6.2.2). We evaluate the baseline centralized insecure, federated insecure, and proposed federated secure RAG pipelines both end-to-end and regarding retrieval-step information leakage on this clinical trend dataset (see Section 6). Finally, we contribute a case study analysis of a decentralized three hospital federated secure RAG system supporting QA on MIMIC-IV anonymized patient EHR records [15], with particular emphasis on **federated retrieval behavior under different user privilege conditions** (see Section 5).

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 2

Related Work

2.1 Clinical-domain LLMs

We summarize prior work on applying natural-language models to the clinical-domain:

ClinicalBERT contextualized word embeddings [2] were obtained by training the general BERT transformer model [9] on both clinical text and discharge summaries. ClinicalBERT achieves better performance than previous medical embedding models and general-domain BERT on the MedNLI [26] natural language inference task from EHR documents. Thus, we plan to use similarity scores between the ClinicalBERT embeddings of source patient EHR documents and a user query to retrieve the top k most relevant documents to a query.

Med-PaLM [31] is the clinical-domain specific instruction prompt-tuned version of Flan-PaLM (an instruction-tuned PaLM LLM), and has demonstrated successful performance on MultiMedQA, a benchmark evaluating LLM performance on medical QA tasks. Med-PaLM answers to medical school knowledge-recall questions achieved 92.6% alignment with scientific consensus as judged by human clinicians, which is comparable to clinicians (92.9%) and much better than the general-domain Flan-PaLM (61.9%). However, the MultiMedQA benchmark and the Med-PaLM language model focus on medical school-type factual knowledge questions that can be answered generally without contextualizing on individual patient medical data. On the other hand, our work attempts the different problem of drawing clinical trends from personal patient EHRs, which necessitates a different model architecture from Med-PaLM.

Almanac [37] uses RAG with the `text-davinci-003` LLM, demonstrating better performance than ChatGPT along the axes of response factuality and completeness using the ClinicalQA benchmark, which comprises medical fact questions such as diagnosis prediction from a description of risk factors and symptoms. Almanac also demonstrated much higher safety against adversarial prompts than ChatGPT (95% of human clinician evaluators judged no intentional harm in response content vs. 0% for ChatGPT). While Almanac also evaluated response accuracy for single-patient diagnosis questions rather than our emphasis on specifically cross-patient/cross-hospital trends, we are interested in exploring its adversarial prompt evaluation framework for assessing the safety of our security-enhanced RAG model.

Ahsan et. al. [1] used the instruction-tuned Flan-T5 XXL LLM to infer patient diagnoses in a zero-shot setting to aid radiologists with unstructured EHR information when making imaging diagnoses. The LLM model was evaluated with human radiologists on the publicly available MIMIC-IV dataset [15], which inspired us to use the updated MIMIC-IV dataset [15] to evaluate our secure-federated RAG model. Ahsan et. al. found that the Flan-T5 LLM approach produced responses consistently preferred over the ClinicalBERT baseline; however, the LLM's major outstanding drawback was its tendency to hallucinate inaccurate evidence (average 9.4% of evidence was hallucinated). The less-preferred ClinicalBERT baseline did not hallucinate since ClinicalBERT is an extractive (non-generative) model that directly retrieves relevant patient information from embedding similarity. While the specific application of clinical LLMs in Ahsan et. al.'s work on assisting radiologic diagnosis is not the focus of our work, it inspired our choice to include Flan-T5-Large [7] as a candidate LLM for the generation step due to its instruction-tuned capabilities and success in this prior study.

Meditron [5] is a suite of open-source LLMs created by fine-tuning Llama-2 [33] on PubMed abstracts, articles, and other published medical documents for clinical-domain QA tasks. Meditron-70B demonstrated exceptional performance on standard medical QA benchmarks that surpassed GPT-3.5 and the closed-source Med-PaLM

[31] and achieved within 10% of Med-PaLM-2 [32]. Although the medical benchmarks targeted medical exam-style questions rather than identifying lateral multi-patient or multi-document clinical trends, Meditron-70B's impressive performance motivated our decision to use the smaller Meditron-7B LLM as a competing open-source, clinical domain-tuned LLM candidate in addition to GPT-3.5-Turbo and Flan-T5-Large during the RAG generation step.

More broadly, there have been many previous applications of LLMs and retrieval-augmented LLMs on medical textbooks (LLM-AMT [35]) or clinical lung-cancer diagnostic tests ([36]) for question-answering tasks in the healthcare domain. However, in summary, prior research largely assumes a central medical knowledge base, which may not be realistic for a distributed set of hospitals. We have not found much relevant prior work on our specific NLP QA task of answering questions regarding clinical trends present laterally across multiple patients, hospitals, or other medical groups of interest.

2.2 EHR Access Control Models

Access and retrieval of patient EHR records demand strict privacy and security guarantees due to the sensitive nature of individualized medical information, which motivated many prior security models on EHR access control [8].

Our architecture uses **policy/attribute-based access control (PBAC)**, which is a standard access control paradigm that has been extensively used across various domains [6]. PBAC frameworks allow for significant flexibility in defining attributes for the users, resources, and environment context impacting access decisions as well as composing policy rules predicated on subsets of those attributes to determine allow/deny decisions. This flexibility makes PBAC a very suitable option for access control over clinical patient EHR documents during our RAG retrieval process since access is often dependent on various user (i.e. role, organization, etc.), document (i.e. department location, whether it contains sensitive information, etc.), and potentially environment (i.e. during working hours, over hospital private network) attributes.

Some prior attribute-based access control models have used the extensible access control markup language (**XACML**) to define access control policies and optionally obligations to be enforced on user accesses to cloud EHR data [29]. On successful authorization, the model additionally uses XML encryption to partially encrypt the EHR document against leakage of sensitive patient data since the primary use case is for very general medical applications in contrast to our work's focus on medical professionals querying information about their specific patients.

Role-based access authorization paradigms in clinical settings has also been explored due to its convenient applicability via mapping to existing hospital roles[8]. Particularly of note is the focus of prior work on emergency fail-safes to bypass RBAC security limits and retrieve patient EHR data if it is pertinent to their survival, otherwise known as "break the glass" [8][3][11]. Ardagna et. al. presents an **exceptions-based RBAC paradigm** [3] that defines policy spaces $\mathcal{P}^+$ ("common practice" positive authorizations), $\mathcal{P}^-$ ("common practice" denied accesses), ε^P (planned exceptions), and ε^U (unplanned exceptions). Space $\mathcal{P}^+$ can be bypassed by policies in ε^U to permit access (requiring an audit log entry) in emergencies. To prevent abuse of this mechanism, policies in space $\mathcal{P}^-$ strictly deny unauthorized accesses with no exceptions. **BTG-RBAC** is a similar paradigm for augmenting existing RBAC infrastructure with a third BTG option (in addition to grant/deny) specifying that a user is able to override unauthorized accesses in emergencies with the caveat that the BTG access will trigger obligations [11]. Our goal is to augment federated RAG to support protecting EHR data with arbitrary access control policies; thus, while we anticipate only demonstrating standard policy-based access control in our implementation, the policy language automatically includes environmental `context` attributes that can be used to define additional break-the-glass access control exceptions within each retriever. Thus, while both the exceptions-based RBAC model [3] and BTG-RBAC [11] are extensions to the role-based access control framework, we elect to use policy-based access control due to its flexibility in supporting traditional RBAC as well as arbitrary "special-case" access exceptions like those proposed without necessitating modifications to the underlying access control paradigm.

Prior work has also explored implementing secure RBAC models on mobile devices accessing healthcare data [28]. On an external HTTP(s) request from a patient, the central web server generates an encoded session key for the patient's smartphone to decode and send to the Extensible Messaging and Presence Protocol (XMPP) server responsible for validating the patient's identity. Upon authentication, the service will query the healthcare institution for accessible EHR data. While our contribution is a web rather than a mobile application, we take inspiration in this work's modularization of the XMPP server and session key encoding on patient requests.

For the general problem of ensuring privacy guarantees in retrieval augmented-LLM generation, **Private Retrieval Augmented Generation (PRAG)** [39] is a recently proposed method for secure retrieval of top-k documents using multiparty computation over a federated retrieval database with $< 1\%$ accuracy loss. A secret-shared inverted file index (IVF) constructed from multi-server federated vector stores is used to perform approximate search while keeping query and document embeddings private. PRAG assumes a flatly-distributed structure of decentralized retriever vector stores while our federated RAG model employs a hierarchical retrieval system; in addition, PRAG presents a privacy-preserving method of federated document-retrieval over *all* server documents without addressing a method of *restricting* access to certain documents via user privileges. The primary focus of our work is the separate problem of securely *excluding* unauthorized documents from the retrieval process entirely, although we believe that privacy-preserving frameworks like PRAG and other differential privacy methods embody promising areas of future research for decentralized retrieval of the remaining access-granted documents under strong privacy guarantees.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 3

Methods

We present our security enhancements to federated RAG frameworks used for question-answering tasks on general cross-clinical document trends, with specific emphasis on integrating **policy-based access control (PBAC) security filters** to guard sensitive patient EHR records during the decentralized retrieval step. We intend our design to securely authorize access for loosely connected systems of independent hospitals which have mutually agreed to **selectively share their knowledge bases** to better inform RAG responses on multi-EHR clinical trends.

The proposed architecture adapts the retrieval module of end-to-end federated RAG pipelines such that each organization or sub-organization department in the federated hierarchy independently defines a set of authorization policies predicated on user role attributes. The retrieved relevant document set is restricted locally via policy access grant/deny decisions determined from specific user attributes so no unauthorized data is exposed to any component's parent retriever(s). Importantly, the final set of documents retrieved as context for the second generation step in the federated secure RAG architecture should only contain documents the authenticated user is authorized to access.

We note that the proposed architecture guarantees a high degree of modularity due to:

- *Independent retrieval and generation steps inherent to RAG.* Retrieval/embedding models can be swapped independently from the LLM generator.

- *Separated authorization security layer from the underlying federated retrieval backbone.* This allows for easy adoption of future document retrieval enhancements, such as EHR knowledge graphs [19] [34], with minimal collateral revisions to the authorization framework, and vice versa.

- *No globally enforced authorization mechanism across organizations* (note: authentication is still performed centrally, as explained in Section 3.2.2). Different retrievers define authorization policies independently and additionally have the flexibility to use a different paradigm entirely (i.e. role-based access control) without affecting other retrievers.

We propose extending the federated insecure RAG (see Section 3.1) architecture to (i) **federated secure RAG** (**FS-RAG**, see Section 3.2), a security-aware RAG model facilitating authenticated PBAC-gated access during federated RAG document retrieval and (ii) an **end-to-end Web application** answering clinical trend queries from a simulated decentralized system of EHR databases while implementing authentication and retrieval authorization policies according to the FS-RAG framework using Python packages `authlib` and `py-abac`, respectively.

3.1 Federated Insecure RAG (FI-RAG)

We briefly sketch the federated RAG architecture serving as the baseline insecure retrieval system, referred to as FI-RAG, in Figure 3-1. The proposed authentication and distributed policy-based access control security model are encapsulated in a separate security layer and are thus generalizable for any retrieval system architecture. However, due to this project's focus on clinical QA, since sensitive patient EHR data stores are often decentralized across different hospital institutions, each of which may have further data isolation boundaries internally, such as between different departments, we adopt an intuitive **tree-structured hierarchy** for the federated retrieval step.

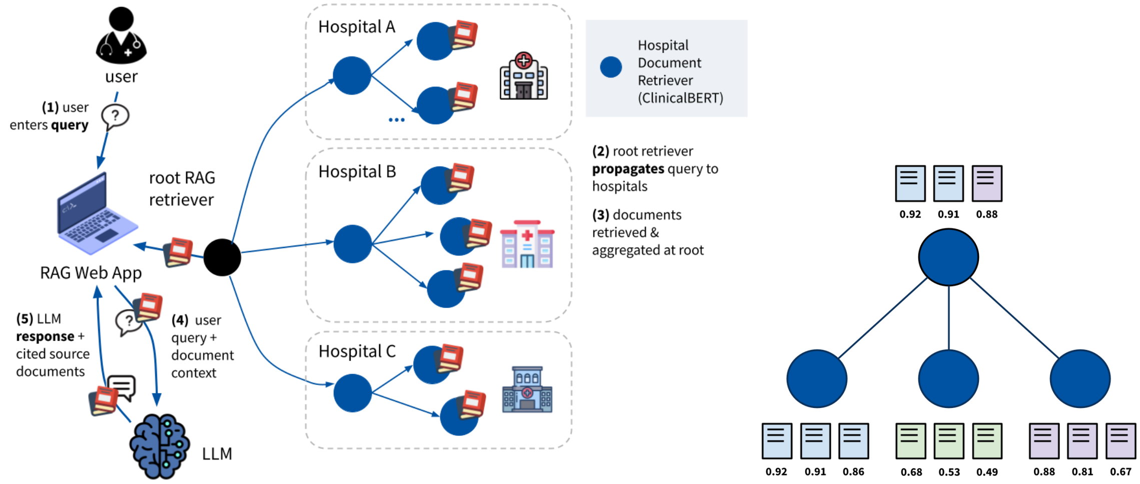

Figure 3-1: Left: Federated Insecure RAG System Architecture. The user's `query` is propagated across the decentralized hospital network and down each hospital's retrieval tree. Retrieval of the top k relevant documents, based on ClinicalBERT embedding cosine-similarity, occurs locally at each leaf retriever's EHR knowledge base. **Right:** Sketch of top k relevant document aggregation. Aggregation is performed at each parent retriever up to the root, which appends the global top k documents as context to the `query` for LLM response generation.

Each leaf retriever's knowledge vector store is initialized with sentence-transformer embeddings of its EHR document database, using mean-pooling over the **ClinicalBERT** contextualized word embeddings [2] per document. On a query, cosine-similarity scores between the query's ClinicalBERT sentence embedding and the EHR document's ClinicalBERT sentence embeddings are used to determine the top k-relevant documents retrieved. These k relevant documents and the query are fed collectively into the response generation step, which is performed using OpenAI's pre-trained **`gpt-3.5-turbo-0125`** LLM [4] (see Section 3.1.1) identically to the standard centralized RAG pipeline.

While this work does not focus on the specific federation mechanisms of FI-RAG in detail, it will extensively modify FI-RAG to implement security control measures. Thus, we first present an abbreviated high-level overview of how the FI-RAG pipeline recursively propagates the user's query to all retrievers in the hierarchy (see Figure 3-1):

1. The clinician prompts the system with a query related to clinical trends. *There is no authentication mechanism to verify or reject the user's identity.*

2. The system forwards the query to the designated root retriever of the hierarchy.

3. The root retriever maintains a set of API retriever endpoints representing the decentralized network of hospitals. *There are no security authorization mechanisms, so the query will be sent to and accepted at all hospitals, including those unaffiliated with the clinician.*

4. Each hospital organization has flexibility to decide the granularity to split retrieval over its patient EHR knowledge base(s) (i.e. between departments). If there exists additional federation within a hospital's retriever hierarchy, the query is recursively routed to all leaf retrievers distributed across the hospital space. *There are no security authorization mechanisms, so the query will reach all leaf retrievers regardless of user role attributes.*

5. Patient EHR databases are only stored at leaf retrievers. The top k-relevant documents to the query by embedding similarity across all called leaf retrievers are retrieved and aggregated back up the FI-RAG retrieval hierarchy. *The leaf retrievers do not implement PBAC security, so some documents could expose private patient data the clinician is not authorized to view.* The documents are appended to the query sent to the generation LLM.

6. In the generation step, the LLM composes a response from the query and document context, citing the source documents for further interpretability.

3.1.1 Notes on Model Selection

The FI-RAG modularized architecture allows for swapping in any retrieval embedding model or generation LLM, but we ultimately decided to use **ClinicalBERT** embeddings [2] and OpenAI's **`gpt-3.5-turbo-0125`** LLM [4] for our end-to-end app implementation of FI-RAG (and FS-RAG). We broadly discuss our selection rationale as follows:

- *Retriever*: We primarily considered the medical domain fine-tuned ClinicalBERT embeddings [2] and the general-domain OpenAI `text-embedding-3-small` embeddings. We qualitatively observed more relevant retrieved document sets using the OpenAI embeddings, especially for staying on-topic if given queries mentioning *specific patient names*, where ClinicalBERT retrieved documents related to the query's general clinical concern but from unrelated patients. However, OpenAI embeddings are closed-source, more costly (in comparison to the freely available ClinicalBERT model), and, critically, use a hosted model incompatible with the local, private nature of patient EHR data. We observed that ClinicalBERT's performance for our use case primarily suffered for specific-patient queries, so we prefixed similarity-search retrieval with an Named-Entity Recognition (NER) model step (using Python `spacy`) to extract patient names, if any, from the natural-language query (see Figure 3-4). The vector store limits the top k relevant documents to only those for the specific patient name(s) identified, guaranteeing that all notes match the queried patient even when using ClinicalBERT.

- *Generator*: We considered the open-source Flan-T5-Large LLM [7], the open-source medical domain Meditron-7B LLM [5], and the closed-source GPT-3.5-Turbo LLM [4] (see Section 6.4). We recognize that GPT-3.5-Turbo is a hosted model and thus is unsuitable for privacy-sensitive patient records under HIPAA [2], but we decided to employ it in our proof-of-concept system implementation since we believe that its significantly better quality responses and ability to synthesize information across documents best demonstrates the potential for retrieval-augmented generation (if properly federated and secure) to assist clinicians identify trends across multiple EHRs. *We strongly recommend organizations switch to a private LLM such as the restricted-access Med-PaLM-2 [32] before deployment on real-world patient data.*

FI-RAG successfully federates retrieval across a decentralized hospital network without requiring aggregated storage of data. However, it lacks an interface for EHR document access authorization, an essential mechanism facilitating hospitals to ensure patient medical record security. The insecure RAG pipeline will search through all leaf retriever knowledge bases on every query; in other words, *all* queries to FI-RAG risk exposing data from *all* leaf retrievers *regardless* of user privileges. Stripping sensitive information from EHR databases may alleviate or mitigate security concerns but will also harm the usefulness of the system for genuine clinicians with authorized access. In order to protect the sensitive data contained in the federated system, we propose to extend federated RAG with fine-grained authorization such that, for example, an attending surgeon's query retrieves from the full EHR knowledge base but a medical school student intern's *same* query retrieves from a much more limited subset.

3.2 Federated Secure RAG (FS-RAG)

In this section, we describe **federated secure RAG (FS-RAG)**, an architecture that integrates *user authentication* and *resource authorization* into FI-RAG. We perform user authentication on top of the **OpenID Connect (OIDC)** standard (built on OAuth 2.0) due to its promising security guarantees and ease-of-implementation. We then implement policy-based access control (**PBAC**) to ensure both coarse and fine-grained control of user attribute-based access to private EHRs during decentralized retrieval.

3.2.1 System Architecture

We maintain the original hierarchical retrieval flow (see Section 3.1) for the proposed federated secure RAG architecture (see Figure 3-2) since authorization security is predicated on proper isolation of the federated data. At the start of a session, users must authenticate to the RAG system through any trusted hospital-associated OIDC server. On successful authentication, the OIDC server returns an `userinfo` token specifying the authenticated user's attributes relevant for authorization.

To begin the federated retrieval process, the RAG root retriever performs a `GET` request containing a tuple (`userinfo`, `query`) of the user's auth attributes and original query to all saved hospital retrieval API endpoints. The FS-RAG authorization layer supports 2 broad mechanisms:

- **Coarse-grained filter over retriever entry**. All router (i.e. non-leaf) and leaf retrievers implement an entry policy decision gate that grants/denies user access before continuing with retrieval. Denying access at a router retriever will prevent the user query from reaching any retrievers in the router's subtree, which may be useful if a hospital or department wishes to coarsely reject access from all members of an unaffiliated hospital or all non-medical staff.

- **Fine-grained filter over documents retrieved**. Leaf retrievers additionally implement a final PBAC document filter after relevant documents are retrieved locally, which strips any access-denied documents out of the retrieved set. Document-access policies make grant/deny decisions based on both `userinfo` attributes and document attributes, which ensures data relevance for authorized user groups while protecting against unintended EHR data leakage to unauthorized users.

3.2.2 Authentication: OpenID Connect

All security mechanisms proposed in FS-RAG rely on correct and secure user identity authentication at the initial step. We elect to use the Python `authlib` package to build secure authentication servers via the OAuth 2.0 OIDC framework.

Each hospital has an associated OIDC auth server. To join the RAG federated system, the hospital auth server generates a OIDC client authorizing requests from the RAG system with the `openid` scope. The user will provide the RAG system an URI of an authentication server; if the URI exists in the RAG system's internal list of trusted hospital OIDC server URIs, the RAG OIDC client for that hospital auth server will visit the auth server's authorization URL in order to retrieve both an

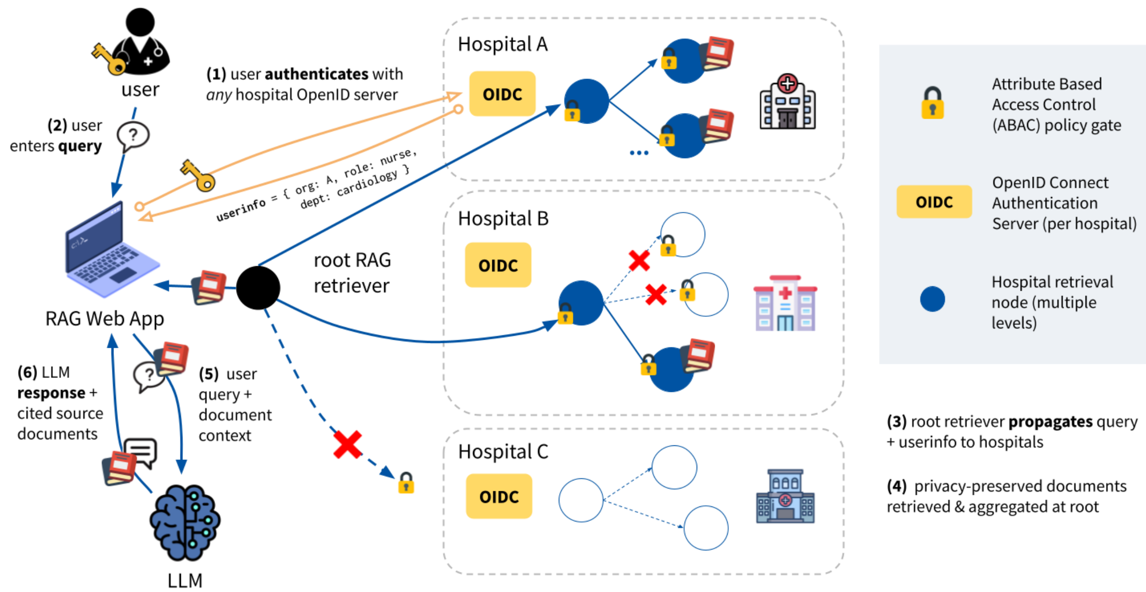

Figure 3-2: Federated Secure RAG System Architecture. Consider an example user, who is a clinician at hospital A with some selective access to hospital B's departments (but none at hospital C). Note that with FI-RAG (Figure 3-1), this user's query will propagate through the entire multi-hospital tree's data. With FS-RAG, the general retrieval-generation pipeline is kept, but the user is allowed access to only hospital A and part of hospital B, guaranteeing no unauthorized documents are exposed at retrieval.

`access_token` and an OpenID `id_token` containing the user attributes `userinfo` (i.e. org, dept, role, etc.) used to perform PBAC policy decisions during retrieval. Before the RAG system is able to retrieve the access and ID tokens, the user is redirected to a hospital-controlled authentication page, where they must successfully log-in with their hospital-affiliated credentials before the OIDC auth server returns the authenticated tokens to the RAG client. The specific method of user authentication is entirely hospital-dependent and outside the scope of our RAG pipeline, since the OIDC server simply needs to redirect to the hospital's authentication portal, which presumably already exists and may include mechanisms like 2-factor authentication, Yubikey, etc. Authentication is considered successfully finished once the RAG retrieval system retrieves the `userinfo` ID token, and only needs to be performed once per session (one session may include multiple user queries).

We allow the user to authenticate through **any** hospital OIDC server in the RAG federation under the assumption that there exists a **baseline level of trust between**

the hospital servers. In other words, our primary use case is a conglomerate system of hospitals with pre-established trust between hospital organizations, rather than a global network across all hospitals carrying the risk of malicious organizations fraudulently authenticating attacker users. We envision FS-RAG as a framework used by multiple local systems of trusted hospitals to secure data access while reaping the benefits of sharing data in a limited and controlled fashion for RAG on clinical trends across the system.

3.2.3 Authorization: Policy-based Access Control (PBAC)

Policy-based access control (PBAC) [6] guarantees flexible document authorization by determining access via a set of attribute-based policies. In contrast to other schemes like role-based access control (RBAC) [27], which only supports defining coarse-grained access privilege levels for pre-defined user groups, PBAC allows for much more fine-grained control over arbitrary user, resource, and typically context attributes. User attributes (i.e. hospital organization, department, medical role, etc. contained in `userinfo`), like in RBAC, are used to precisely define the group of users that a policy $\mathcal{P}$ is defining access for. Resource attributes (i.e. EHR data type, data-sensitivity class, etc.) are used to define the set of resources that a policy is applicable for. Furthermore, context attributes may also be used to differentiate authorization decisions depending on changing environmental attributes, such as time of day (i.e. within hospital working hours) or geographic location.

A PBAC policy is defined as $\mathcal{P}_{R_U,R_R,R_C}(u,r,c,a) \mapsto D$, where

- R_U, R_R, R_C are rule sets over user attributes, resource attributes, and context attributes, respectively, and define the permissible values for each attribute for the specific type of access controlled by $\mathcal{P}$
- $D = \{\texttt{DENY}, \texttt{ALLOW}\}$ is the set of possible policy decisions
- u, r, c represent attributes of the specific user, resource, and context of the evaluated access

- a is the critical action regulated by the policy (i.e. read, write, execute data)

The policy $\mathcal{P}_{R_U,R_R,R_C}$ only applies to user, resource, context attribute tuples (u, r, c) satisfying all attribute-rule sets (i.e. $R_U(u) \wedge R_R(r) \wedge R_C(c) = \mathbf{true}$). For such a policy-applicable tuple of (user, resource, context) attributes, $\mathcal{P}_{R_U,R_R,R_C}(u, r, c, a)$ asserts the allow/deny authorization decision for action a on the resource. If any of the rule sets evaluate to **false** for the given (u, r, c) attributes, the policy does not apply (i.e. it neither denies or allows the access).

Our implementation and PBAC model only defines **allow** policies $\mathcal{P} \mapsto$ ALLOW (i.e. whitelists). We note that the FS-RAG framework itself does not impose such policy-language restrictions, so hospital administrators have flexibility to define deny policies or other policy-evaluation paradigms instead. However, *mixing allow and deny* policies would cause authorization decisions to be dependent on the *order* that policies are evaluated at every PDP. For example, if we consider two policies $\mathcal{P}_1 = \{$ALLOW if department is CARDIOLOGY$\}$ and $\mathcal{P}_2 = \{$DENY if role is NURSE$\}$, it is non-obvious whether a nurse in the cardiology department should be allowed access or not (may be `ALLOW` if P_1 allows access before P_2 is evaluated, or vice versa for `DENY`). This evaluation-order complexity from hospital-specific policy definitions is out of the scope of our generalized PBAC-authorization scheme. It is also possible to define *only deny* policies, but due to the sensitive nature of private EHR data, we argue that users should be denied access by default and selectively granted access, which makes exclusively using deny policies more cumbersome. Errors are also significantly more damaging with deny policies since any user not explicitly denied access would be able to view patient-private medical data; with allow policies, errors cause the more benign inconvenience of unintentionally limiting an authorized user's access.

A **policy decision point (PDP)**, instantiated from a set of policies, is the actual policy evaluation component that determines allow/deny decisions during retrieval. In our PBAC protocol, the PDP returns `ALLOW` iff at least 1 policy evaluates to `ALLOW` (i.e. OR-evaluation). If no policies evaluate to `ALLOW` (i.e. since we only support allow policies, this implies that *no policies* applied to the particular (user, resource, context, action) attribute tuple), the PDP returns `DENY`.

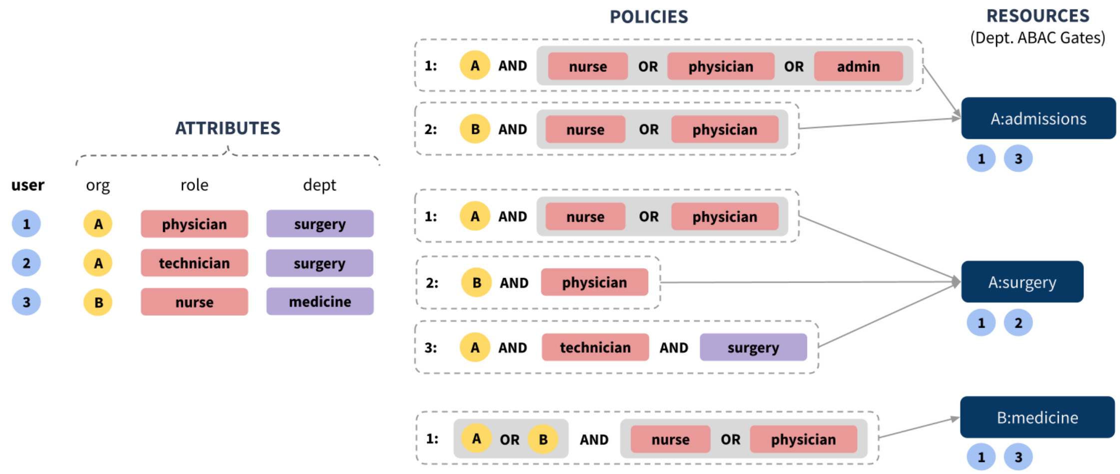

Figure 3-3: Sketch of select PBAC (allow) policies authorizing access to the retrievers (`A:admissions`, `A:surgery`, `B:medicine`) based on user attributes (`org`, `role`, `dept`). Users are granted access if at least one (allow) policy applies.

For a concrete example (see Figure 3-3), consider the user attribute set $U_{\text{ATTR}} = \{ORG, ROLE, DEPT\}$. In the interest of simplicity, we ignore resource and context attributes when defining access policies (i.e. $R_{\text{ATTR}} = \{\}$, $C_{\text{ATTR}} = \{\}$). Each of the `A:admissions`, `A:surgery`, `B:medicine` retrievers independently constructs a local PDP. Some user attribute rules R_U for the policies are translated below (see Figure 3-3 for the full list):

$$R_U^{\texttt{A:adm},1}(u) = (u_{ORG} = \texttt{A}) \wedge ((u_{ROLE} = \texttt{NURSE}) \vee (u_{ROLE} = \texttt{PHYSICIAN}) \vee (u_{ROLE} = \texttt{ADMIN}))$$
$$R_U^{\texttt{A:adm},2}(u) = (u_{ORG} = \texttt{B}) \wedge ((u_{ROLE} = \texttt{NURSE}) \vee (u_{ROLE} = \texttt{PHYSICIAN}))$$
$$\vdots$$
$$R_U^{\texttt{A:sur},3}(u) = (u_{ORG} = \texttt{A}) \wedge (u_{ROLE} = \texttt{TECHNICIAN}) \wedge (u_{DEPT} = \texttt{SURGERY})$$

Since we only implement allow policies, in this simplified example, $\mathcal{P}^i(u) = \texttt{ALLOW}$ for user-attributes u if its associated user-attribute rule $R_U^i(u) = \textbf{true}$. The PDP decision is then `ALLOW` if $\bigvee_i \mathcal{P}^i(u) = \texttt{ALLOW} \implies \bigvee_i R_U^i(u) = \textbf{true}$ (i.e. some allow policy applied) and `DENY` otherwise.

For example, an access request for user 1, who has attributes ($u_{ORG} = \texttt{A}, u_{ROLE} = \texttt{PHYSICIAN}, u_{DEPT} = \texttt{SURGERY}$), to the `A:admissions` PDP will evaluate to `ALLOW` since policy 1 applies and grants access.

Enforcing PBAC with py-abac

We implement our policy access control scheme using the Python package `py-abac`. Each hospital retriever maintains a local policy database P_{gate} controlling retriever-granularity access, and leaf retrievers additionally maintain a local policy database P_{docs} controlling document-granularity access. For our proof-of-concept implementation (see Section 4), we initialized all policy stores using in-memory storage for simplicity, but organizations are able to easily swap to a persistent storage model (i.e. SQL database) on deployment. Each retriever then creates a `py-abac` policy decision point (PDP_{gate}) from the P_{gate} database (and likewise for document-access PDP_{docs} points) configured to `ALLOW_OVERRIDES` to reflect our PBAC paradigm that access is granted if *any* policy decision evaluates to `ALLOW`. Access requests during retrieval are structured as JSON objects containing attribute values for the specific `subject`, `resource`, `action`, and `context` of the resource access operation.

Coarse-grained PDP$_{gate}$ decision flow

Each federated secure RAG retriever exposes a `get_relevant_documents` method that takes in (`userinfo`, `query`) as arguments and returns the top k relevant documents to the `query` across all authorized documents in its retriever subtree. When a retriever's `get_relevant_documents` method is called, the retriever policy decision point PDP$_{gate}$ evaluates the user attributes from the `userinfo` token (i.e. returned by some OIDC server on successful authentication in the previous step) against the retriever policies P_{gate} and determines if the user has access to the current retriever. In the coarse-grained access control filter case, the resource is the retriever itself, so resource attributes include the department (`dept`) whose data the retriever has ownership of, and so on.

The primary alternative design that we considered was to have each retriever handle policy-based access control for entry to each of its child retrievers. In other words, each router (non-leaf) retriever would maintain a list of child retrievers $C_1, \ldots, C_r$ and a set of policies authorizing access to each child retriever $\mathcal{P}_{C_1}, \ldots, \mathcal{P}_{C_r}$. Before recursively calling each child retriever C_i's `get_relevant_documents` method, the router retriever performs a policy access control check with the user attributes `userinfo` and the resource attributes for C_i.

In contrast, our current design **encapsulates all access control authorization logic within each retriever**. Each router retriever (assuming the router's entry PDP_{gate} has authorized access to the general router) will *always* propagate the user query/attributes to *all* of its child retrievers. Each child retriever is responsible for implementing a coarse-grained access allow/deny PBAC-gate at entry, as we described above with the PDP_{gate} flow. We elected to delegate PBAC authorization entirely to each retriever in this fashion since this supports the most flexibility and control for each retriever to update their PDP_{gate} policies *independently* of other retrievers in the federated system. In the alternative design, a child retriever modifying their policies to authorize or take away access for a subset of users would necessitate communicating with the parent router retriever to modify the parent's PDP. In addition, the amount of PBAC policy information stored at each (router) retriever would be $O(r)$, where r is the number of child retrievers, since the parent retriever must handle all PBAC-authorization on behalf of its children, which suffers from a scalability standpoint for hospital networks with many child retrievers per router. Our current encapsulated-PBAC design, on the other hand, stores a constant $O(1)$ amount of PBAC policy information regardless of how many child retrievers are connected to a router retriever, so adding many new child retrievers into the retriever hierarchy does not put any additional PBAC-policy storage burden on existing parent retrievers.

Fine-grained PDP_{docs} decision flow

As in FI-RAG, the actual relevant document retrieval over sensitive patient EHR data is only performed at leaf retrievers. We augment leaf retrievers to additionally perform a final access control filter over the set of relevant documents retrieved $D_{\text{fetch\_k}} \subset D_{KB}$ from all documents in the leaf knowledge base D_{KB} (see Figure 3-4). For a target of k most relevant documents retrieved, we fetch a slight surplus of `fetch_k` $> k$

documents in the initial step since we anticipate that a subset of the most relevant documents may not be authorized for the current user to access. For each document $d_i \in D_{\text{fetch\_k}}$, the leaf document policy decision point PDP_{docs} will determine if the user is allowed access based on the document attributes $d_{i_{ATTR}}$ (i.e. type of data, located in which part of EHR record) and the user attributes in `userinfo`. This results in a set of authorized *and* relevant documents $D_{auth} \subseteq D_{\text{fetch\_k}}$ given by

$$D_{auth} = \{d_i \in D_{\text{fetch\_k}} \mid \text{PDP}_{docs}(\texttt{userinfo}, d_{i_{ATTR}}) = \texttt{ALLOW}\}$$

for the given query and user. If $|D_{auth}| > k$, then we truncate (and then return) D_{auth} to the top k most relevant documents by ClinicalBERT embedding similarity score to the query.

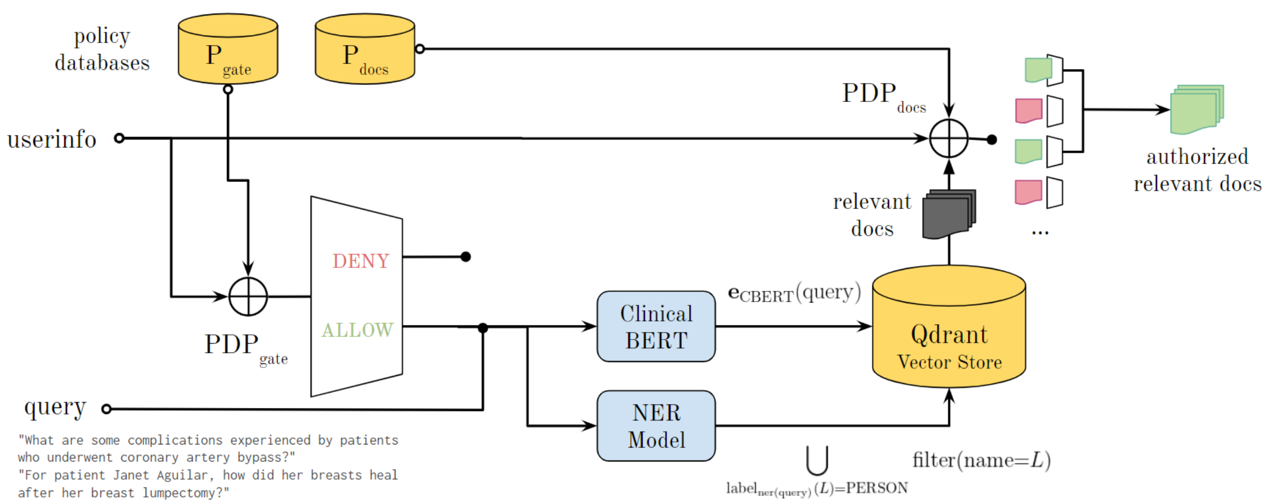

Figure 3-4: Diagram of PBAC-restricted document retrieval at leaf retrievers given `userinfo` attributes and a text `query`. Policy decision points PDP_{gate} and PDP_{docs} are initialized from internal policy storage and grant entry to the retriever or access to final document data, respectively, if *any* policy returns "ALLOW". Otherwise, access is denied.

For implementing leaf KB document retrieval, we adopted the **Qdrant** vector store framework rather than FAISS [18] for our use case primarily because Qdrant's support for conditional filtering during retrieval similarity search and data management (including data backup/restore) are better suited for the dynamic nature of medical EHR records. FAISS is a static vector store with optimizations for fast/efficient and accurate similarity search, but it does not support filtering during search (with the exception of some workarounds that would undermine the efficiency of FAISS similarity search), so we would only be able to perform the patient-name(s) filter (from the `spacy` NER model labels created in a pre-similarity search step on specific-patient queries), or any future filter extensions to the retrieval system based on hospital needs, **after** the FAISS similarity search over all D_{KB} has finished. Using Qdrant, embedding similarity search can be performed on a slice of the D_{KB} vector store index filtered on patient name.

While we perform the patient-name(s) filter *during* similarity-search, the document PDP_{docs}-controlled authorization filter is performed *afterwards* on the retrieved document set $D_{\text{fetch\_k}}$. On a practical level, Qdrant does not support variable search-filters at this complexity for good reason: similarity-search performance would greatly suffer as the filter requires a PDP_{docs} policies evaluation decision on all $|D_{KB}|$ documents that cannot be precomputed (unlike static name filters) since the access decision is dependent on the user's attributes as well. Our design to perform the filter post-similarity search does leave the possibility for $|D_{auth}| < k$ (less than k documents retrieved) if the user has authorized access to only a very small subset of documents in the most relevant fetched set $D_{\text{fetch\_k}}$. However, we argue that in a situation where the user does not have access to the majority (or any) of the best relevant documents, it is preferable (to prevent LLM hallucinations) to send the LLM generator a smaller but highly relevant document subset rather than include irrelevant documents, provided that `fetch_k` is set to an appropriately large value (our default is 20 for a k value of 10).

3.2.4 Authorization Alternatives

Though we initially planned on incorporating role-based access control (RBAC) [27] authorization, we ultimately adopted policy/attribute-based access control as PBAC's flexibility is better suited to the *dynamic* and *decentralized* nature of hospital EHR data and organizations:

- Hospital staff organization and patient EHR data **change frequently**. RBAC is most suited to *static* roles and permissions determined before system deployment, and makes role-permission modification much more difficult than a policy-based approach, in which authorization policies in the P_{gate} or P_{docs} databases can be flexibly updated on the same set of user/resource attributes.

- FS-RAG involves controlled, collaborative data sharing between a distributed system of hospitals, so **federated authorization enforcement is complicated by varying role definitions and permissions across organizations.** A PBAC-secured retriever at hospital A can define policies allowing similarly restricted read access for nurses at A and physicians at an external hospital B. An RBAC-secured retriever, however, would require an unnatural role-mapping between external roles to internal roles (i.e. including unintuitive translations like "physicians at B $\rightarrow$ nurses at A").

- PBAC fine-grained policy language allows hospital resource owners to **tailor specific access permissions(s) to a precise user-resource combination** (and possibly environment context). To mimic this effect, static RBAC requires a new role for each collection of user-resource permissions, which may result in RBAC system "role explosion" [20].

Chapter 4

Implementation

We implemented the core federated RAG logic (see Section 3) using **LangChain**, a framework facilitating custom creation of LLM and RAG question-answering retrieval pipelines (through the Python `langchain` package) [21]. We then implemented security extensions to the federated retrieval step using the Python libraries **`authlib`** to build OIDC authentication servers and **`py-abac`** to administer policy-based access control during retrieval (see Section 3.2.3).

Our contribution is a flexible Python library framework to streamline building **per-hospital federated and secure RAG modules**, which we demonstrate in our 3-hospital case study implementation (see Section 5). We also constructed an user-facing **end-to-end RAG Flask web application** that answers clinical trend queries from authenticated users while guaranteeing that the cited documents and responses draw only from user-accessible EHR data. The end-to-end RAG app communicates with the hospital Flask servers using HTTP GET and POST requests in order to authenticate users and perform federated retrieval across the distributed retrieval network. Since our case study implementation is a proof-of-concept to demonstrate the viability of the FS-RAG architecture, we have all 3 hospital Flask servers and the central web app listening to different ports (i.e. `5001`, `5002`, `5003` for the hospitals and `5000` for the central app) on the same OpenStack VM instance. While future work on real-world deployment necessitates a more sophisticated implementation on separate physical machines, we believe that our design adequately demonstrates the core federation and security principles of our model since the servers are functionally isolated in different processes and listening on different ports (i.e. they are not sharing EHR data beyond the intentional specifications of our federated retrieval process).

4.1 OIDC Authentication Server

Our goal is to delegate as much control to the hospital organizations as possible while preserving the functionality of the RAG pipeline so that hospitals can independently define security rules best suiting their organizations. Along these lines, the central RAG web app redirects all user authentication logic to the hospital servers; in other words, users must successfully authenticate with any hospital in the federation, which will then inform the central RAG app of the user's identity.

Each FS-RAG hospital Flask server has a high degree of flexibility in specifying the user authentication process. Since the exact method of user authentication is not the focus of our model, our implementation simply uses a username login field, but hospitals are free to employ two-factor authentication and other more secure methods.

Minimally, we require each hospital server to implement functionality for serving as an **OIDC Authentication Provider**. To streamline this process, we publish scaffolding code in `orgs/auth_skeleton/` of the codebase and outline the process to enable OIDC authentication for a new hospital joining the federation below.

A hospital Flask app (i.e. see `orgs/hospitalA/app.py` for a concrete example) should invoke the `setup_auth` method with the configuration information of the hospital, including the `OAUTH2_JWT_ISS` (i.e. hospital server URI), `OAUTH2_JWT_ALG`, and other fields. We note that our proof-of-concept OIDC implementation utilizes symmetric key encryption (HS256) for simplicity, but we recommend hospitals use an asymmetric encryption method for deployment (i.e. RS256). The `setup_auth` method will handle setting up the OIDC database tables and exposing the necessary authentication Flask endpoints.

It is not necessary for hospitals to modify any `auth_skeleton` code to create a functioning OIDC provider, but we briefly describe the key components of the framework for background context. We summarize the Flask endpoints for handling OIDC authentication (see `orgs/auth_skeleton/oidc_bp.py`) in Table 4.1.

Table 4.1: OIDC Authentication Flask endpoints exposed by each hospital server.

Endpoint	**Description**
`/oidc/login`	Logs in to hospital server with user credentials.
`/oidc/logout`	Logs user out of hospital server.
`/oidc/create_client`	Creates OIDC `client_id`, `client_secret` pair associated with an application. The central RAG app must invoke this endpoint once for every hospital to create a valid RAG app client handling authentication for that hospital.
`/oidc/authorize`	Authenticates the user and then prompts the user to share userinfo data with the central RAG app. On agreement, issues a valid OIDC `authorization_code`.
`/oidc/token`	Validates request. If there is a valid `authorization_code`, issues `access_token` and `id_token` (userinfo)
`/oidc/revoke`	Validates request and marks `access_token` as revoked.

We also create the database tables `OpenIDClient`, `OpenIDAuthorizationCode`, `OpenIDToken` to store valid client information, authorization codes, and issued auth tokens by the hospital OIDC provider, respectively. Finally, when the hospital server calls `setup_auth`, the skeleton code will configure the server as an OIDC authentication provider by registering an `AuthorizationCodeGrant` with the `OpenIDCode` claim that will then handle issuing/denying access and user ID tokens on authentication.

4.2 PBAC Authorization

To communicate with the central RAG app during the federated retrieval, each hospital server exposes an additional endpoint at `/api/retrieve` (see Listing 4.1). The internal hospital document retrieval process is invoked via the `get_relevant_documents` call at lines 7-9, where `org_retriever` is a `RouterRetriever` node object for the hospital organization that stays in scope for the entire lifetime of the Flask server.

Listing 4.1: Hospital PBAC-enabled retriever API endpoint.

```
1  @hosp_bp.route('/api/retrieve', methods=['GET'])
2  def retrieve():
3      if request.method == 'GET':
4          query = request.args.get('query')
5          userinfo = json.loads(request.args.get('userinfo'))
6          search_kwargs = json.loads(request.args.get('search_kwargs'))
7          docs = org_retriever.get_relevant_documents(query=query,
8                                                      userinfo=userinfo,
9                                                      search_kwargs=search_kwargs)
10         return jsonify(docs=[doc.to_json() for doc in docs], query=query)
11     return jsonify()
```

We implemented the FS-RAG `RouterRetriever` and `LeafRetriever` classes as specialized LangChain `Retriever`'s (which they both inherit from) that handle federating document retrieval down a hierarchy while regulating retrieval access via both coarse-grained retrieval-level and fine-grained document-level PBAC policy decision points. Hospitals must make at least 1 root `RouterRetriever` that is called at the `/api/retrieve` API endpoint but may additionally create arbitrarily more in the case of a more complex retrieval organization or demand for more department access control levels. Hospitals store all EHR records at its terminal `LeafRetriever` nodes, which are responsible for performing the actual document retrieval and PBAC-filtering given the user's original `query` and `userinfo` auth attributes.

At all `RouterRetriever` and `LeafRetriever` nodes, we support defining `py-abac` PBAC policies in JSON format (see Listing 4.2 for an example from our case study). Each policy must specify an `effect` (which we default to `allow` for our use case since we only consider allow policies as described in Section 3.2) and a set of `rules` used to specify the subset of users (`subject`), resources (`resource`), and environmental conditions (`context`) whose access (as specified in the `action` field) is governed (i.e. allowed) by the policy.

Listing 4.2: Example py-abac JSON policy definition gating A's orthopaedics retriever.

```
{
  "uid": "1",
```

```
  "description": "Allow orthopaedics access to hospital A radiology technicians",
  "effect": "allow",
  "rules": {
    "subject": {
                "$.org": {"condition": "Equals", "value": "A"},
                "$.role": {"condition": "Equals", "value": "technician"},
                "$.dept": {"condition": "Equals", "value": "radiology"}
              },
    "resource": {"$.dept_id": {"condition": "Equals", "value": "orthopaedics"}},
    "action": [{"$.method": {"condition": "Equals", "value": "read"}}],
    "context": {}
  },
  "targets": {},
  "priority": 0
}
```

For example, in Listing 4.2, the `subject` rule block indicates that the user attributes should have `org` equal to `A` AND `role` equal to `technician` AND `dept` equal to `radiology`.

Hospitals define lists of such policies to control entry access to each retriever (i.e. PDP_{gate}) and to control document-level access after leaf retrieval (i.e. PDP_{docs}). For our case study (see Section 5), each hospital has 1 `RouterRetriever` and a set of department-level `LeafRetriever` nodes. They additionally define static dictionaries `DEPT_GATE_POLICIES` and `DOC_POLICIES` keyed by department name, where all PDP_{gate} and PDP_{docs} policies for each `dept` are specified in the lists `DEPT_GATE_POLICIES[dept]` and `DOC_POLICIES[dept]`, respectively. From these policy stores, the hospital can create retriever gate-level and document-level PDPs for its department `LeafRetriever` nodes as detailed in Listing 4.3. Note that we specify `EvaluationAlgorithm.ALLOW_OVERRIDES` as the PDP algorithm so that access is granted at a PDP gate **if and and only if at least 1** such allow policy in a retriever's PDP policy set applies to the user with attributes `userinfo` and the requested resource.

Listing 4.3: Setting up PDP objects for all department leaf retrievers at a hospital.

```python
for dept in DEPTS:
    dept_gate_policies = [Policy.from_json(policy_json) for policy_json
                                                        in DEPT_GATE_POLICIES[dept]]
    storage_gate = MemoryStorage()
    for policy in dept_gate_policies:
        storage_gate.add(policy)
```

```
    pdp_gate = PDP(storage_gate, EvaluationAlgorithm.ALLOW_OVERRIDES)

    doc_policies = [Policy.from_json(policy_json)
                         for policy_json in DOC_POLICIES[dept]]
    storage = MemoryStorage()
    for policy in doc_policies:
        storage.add(policy)
    pdp_docs = PDP(storage, EvaluationAlgorithm.ALLOW_OVERRIDES)

    dept_retriever = LeafRetriever(id=dept,
                                pdp_gate=pdp_gate,
                                pdp_docs=pdp_docs,
                                attrs={'org': 'A', 'dept_id': dept})
    child_retrievers.append(dept_retriever)
```

A similar process is performed to create the `pdp_gate` PDP regulating access to the hospital-level `RouterRetriever` that is initially called at the start of retrieval (see the Flask endpoint in Listing 4.1).

`gate_req_json` in Listing 4.4 is an example of a formatted PBAC access request generated at the initial gate access check of every `LeafRetriever`'s `_get_relevant_documents` method (the initial underscore indicates that the leaf method is internal to the hospital hierarchy and should not be called at external-facing Flask endpoints). This particular access request is targeting the retriever's gate PDP, but requests to the document PDP would follow the same JSON format with the `resource` field replaced with the document `metadata` attributes. In the example gate JSON request, the user/subject attributes (i.e. `org`, `dept`, `role`, `affiliations` for our case study, but our framework supports any) are obtained from the passed-in `userinfo` dictionary. Since the gate PDP regulates user entry into the retriever, the `resource` field is set to metadata describing the retriever (i.e. the department name). The `action` field is always set to `read` in our FS-RAG implementation since we are only concerned with retriever/document read access during RAG retrieval, and the user should never be allowed write or execute privileges.

Listing 4.4: Abbreviated PBAC-authorized document retrieval in leaf retrievers.

```
def _get_relevant_documents(
    self, query: str, search_kwargs: dict, userinfo: dict = None, **kwargs
) -> List[Document]:
    gate_req_json = {
```

```
        "subject": {
            "id": userinfo.get('sub') if userinfo else None,
            "attributes": userinfo or {},
        },
        "resource": {
            "id": self.id,
            "attributes": self.metadata or None,
        },
        "action": {
            "id": "",
            "attributes": {"method": "read"}
        },
        "context": {}
    }
    # First authorize coarse-grained retriever access via PDP_gate
    if not self.pdp_gate.is_allowed(AccessRequest.from_json(gate_req_json)):
        return []
    # Retriever-access authorized. Retrieval progresses onto Qdrant
    # embedding similarity search below...(truncated for brevity).
```

The `userinfo` dictionary is propagated from the central RAG app, which obtained it from a valid hospital's OIDC provider upon successful user authentication, to each hospital's internal retrieval hierarchy as part of the `args` field in the HTTP GET request sent to the hospital's exposed `/api/retrieve` endpoint. The endpoint extracts the `userinfo` dictionary and `query` string and passes both to the internal hospital `RouterRetriever`, which continues to propagate them down when invoking the `_get_relevant_documents` method of any subtree child retrievers. The `userinfo` dictionary is used to evaluate user access for entry into retrievers as seen in Listing 4.4, as well as user access to the final retrieved document set, in which a JSON PBAC request is constructed for each of the `fetch_k` retrieved documents from the leaf Qdrant knowledge base and used to filter out unauthorized documents before returning.

4.3 RAG Web Application

The end-to-end central RAG web application is implemented as a Flask server communicating with the hospital servers over HTTP during the retrieval step. On start up, the RAG app configures an `OAuth` client and registers all known trusted hospital

OIDC authentication providers in the federation (see Listing 4.5).

Listing 4.5: Registering hospital OIDC servers with the RAG app OAuth client.

```
def setup_auth(app):
    oauth = OAuth(app)
    for server_uri, server_dict in TRUSTED_OPENID_SERVERS.items():
        oauth.register(
            name=server_dict["name"],
            server_metadata_url=server_dict["server_metadata_url"],
            jwks={
                "keys": [{
                    "kty": "oct",
                    "alg": "HS256",
                    "k": urlsafe_b64encode(str.encode("secret-key")),
                }]
            },
            client_kwargs={
                'scope': 'openid profile'
            }
        )
    return oauth
```

The `TRUSTED_OPENID_SERVERS` dictionary is extracted from a config file at the RAG app that specifies each hospital Flask server URI and other important metadata information. Since our proof-of-concept implementation uses the simpler symmetric HS256 encryption algorithm for demonstration purposes, the OAuth client must additionally provide a valid `jwks` key for each server. We suggest using the asymmetric RS256 encryption algorithm when extending this implementation for deployment, in which case the client would not need to provide a shared key. The `server_metadata_url` fields are hospital server-exposed endpoints (i.e. http://127.0.0.1:5001/.well-known/openid-configuration for Hospital A in our case study) that serve static JSON objects pointing to the authorization, token creation, and other authentication-relevant URLs for each hospital's OIDC server.

We outline the user-facing authentication flow starting at the front page of the central RAG app (see Figure 4-1). The user will provide an user-owned URI to the RAG app, which will dereference the URI via an HTTP GET request. The user-owned URI response should be a JSON-formatted dictionary containing the key `openid_server_uri`, which should point to an hospital OIDC authentication server

Figure 4-1: Screenshot of front page of RAG web application. To begin querying the system, the user must first authenticate with a trusted hospital OIDC server. To do so, the user provides a self-owned URI as depicted in the screenshot, which should point to a trusted OIDC server URI when dereferenced.

URI affiliated with the user. If the RAG app validates the provided hospital OIDC URI against its private list of `TRUSTED_OPENID_SERVERS`, it will redirect to the authorization endpoint for the OIDC server, which the RAG app can obtain from its OAuth client since it registered the server metadata URLs of all trusted OIDC servers in the prior startup step (see the `/login` endpoint in Listing 4.6). The `getattr(oauth, oidc_server)` command is necessary to select the specific hospital OIDC provider whose URI the current user provided for authentication among the RAG app's registry of all trusted servers. During this authentication step, control flow is redirected from the central RAG app to the `/oidc/authorize` endpoint exposed by the hospital authentication server, which is free to implement any user authentication method in practice (see Section 4.1). Figure 4-2 is a screenshot of the user-facing interface immediately after a valid user authenticates with the hospital OIDC provider. The authenticated user is prompted to consent to sharing their OpenID `userinfo` scope information with the central RAG application, including their hospital-affiliated department, role, and other identity attributes.

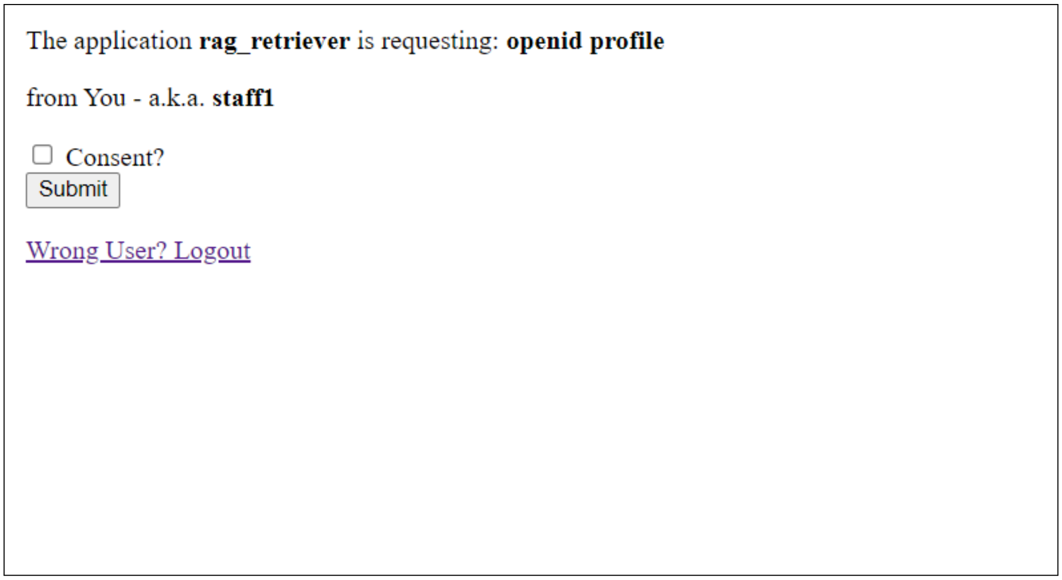

Figure 4-2: Screenshot of web application interface at the `/oidc/authorize` endpoint of a hospital server during user authentication.

On successful authentication and consent, control is returned to the central RAG app, which is redirected to the `/auth` endpoint (see Listing 4.6). The RAG app then invokes the `authorize_access_token` method of the selected OIDC server, which will validate the token request and authorization code (obtained at the previous step) and return authorized access/ID tokens to the RAG app.

Listing 4.6: Central RAG app endpoints to handle initial user authentication.

```
@app.route('/login')
def login():
    oidc_server = session['oidc_server'] if 'oidc_server' in session else None
    assert oidc_server is not None

    redirect_uri = url_for('auth', _external=True)
    return getattr(oauth, oidc_server).authorize_redirect(redirect_uri)

@app.route('/auth')
def auth():
    global rag_chain_with_source
    oidc_server = session['oidc_server'] if 'oidc_server' in session else None
    assert oidc_server is not None

    token = getattr(oauth, oidc_server).authorize_access_token()
    session['user'] = token['userinfo']
    rag_chain_with_source = create_rag_chain_with_source(session['user'])
    return redirect('/success')
```

Using the issued authorized `token` information given by the OIDC authentication server, the central RAG app saves the user attributes from `token['userinfo']`, which it will need to submit along with the user's `query` to later perform access control protected retrieval. Then on line 17 of Listing 4.6, the RAG app constructs a RAG retrieval chain that performs federated secure retrieval as described in the earlier Section 3.2 with the logged-in user's attributes. The chain is composed of the following components:

1. **RootRetriever**: The central root RAG retriever initiates the retrieval step across the federated hierarchy of hospital retrievers. The retriever is initialized with a list of all connected hospital `/api/retrieve` endpoint URIs, and will serially send an HTTP GET request to all endpoints with the user's original `query` and their security `userinfo` attributes. In the case that a hospital server is down or the HTTP request errors out for another reason, the `RootRetriever` suppresses the error and moves on to the other hospitals in order to avoid crashing the entire application due to a single hospital failure.

2. **LLM Prompt**: The retrieved document context and query are combined into a final formatted prompt that is fed into the generation step. We discuss prompt selection in-depth later in this section.

3. **Response Generation**: The generation step is identical to the centralized RAG pipeline (or to any non-RAG extended LLM) since a driving motivation of retrieval-augmented generation systems is to encapsulate query context retrieval in a prior step (and for FS-RAG, the federation and access control frameworks as well) to avoid the cost of fine-tuning LLMs. As also discussed in Methods, we used `gpt-3.5-turbo` [4] with parameter `temperature = 0` to reduce generation LLM noise (we cannot completely eliminate noise since GPT-3.5 is not 100% deterministic) when observing the impact of our retrieval modifications on the produced responses.

The prompt template used to format the original query (`question`) and retrieved documents (`context`) into a single string input passed to the GPT-3.5 LLM is reproduced in Listing 4.7.

Listing 4.7: LLM prompt template formatting the query and retrieved documents.

```
TEMPLATE = """You are an assistant for clinical question answering.
          Read the following clinical note excerpts:
          {context}
          The above notes may not all be relevant. If no notes were provided,
          or there is insufficient information to answer, please say so without
          stating false information. Please answer in your own words,
          providing evidence from the relevant notes:
          {question}"""
PROMPT = ChatPromptTemplate.from_template(TEMPLATE)
```

We experimented with varying prompt instructions and inspected the generated responses for document **citations**, **synthesized conclusions** from multiple documents (rather than quotations), and **acknowledgement of lack of information** rather than hallucinations, especially for users with limited EHR access. We found that including the user's query as the final statement of the prompt, after the instructions and document context, produced more relevant responses to the content of the query, possibly due to recency bias for the query in the LLM context window when stated last. The statement that "*the above notes may not all be relevant*" was inserted to suggest ignoring irrelevant statements after we observed that the LLM attempted to extract information from all documents, some of which were not relevant. We also included the instruction to state "*insufficient information to answer*" when there was inadequate retrieved information rather than fabricating a response in hopes of reducing hallucinations. As we discuss in Section 6, the majority of responses to users with no relevant EHR access correctly mention insufficient information, although there were still a few instances of hallucinations.

All authorization filters occur at the retrieval step, so unauthorized documents will never be exposed to the generation LLM at all. This is preferable to an alternative design that feeds all document context into the LLM but tunes the prompt such that the LLM restricts the context cited in the generated response, since implementing access control at the generation LLM step leaves the system vulnerable to adversarial prompting and leakage of unauthorized EHR data on hallucinations. FS-RAG is still susceptible to generation hallucinations, but since the generator is never exposed to unauthorized data, hallucinations are simply fabrications, which are still problematic and misleading, but will not inadvertently leak patient data.

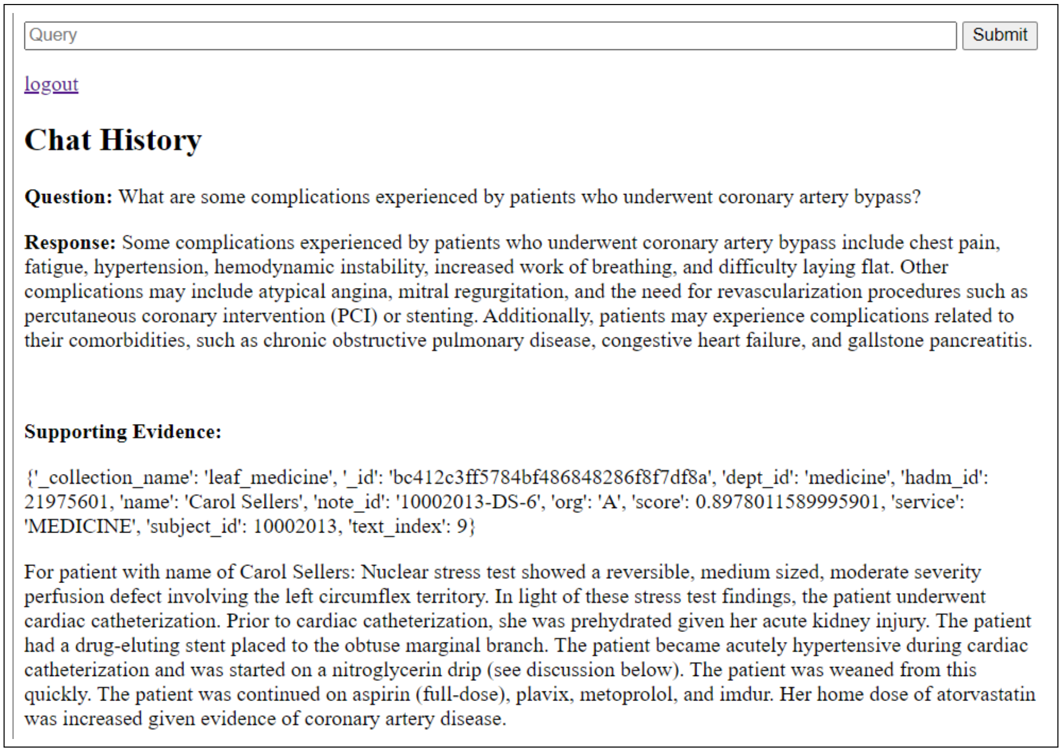

Figure 4-3: Screenshot of a portion of the RAG web application on successful end-to-end execution of the federated/PBAC-secured pipeline. The original query, generated response, and retrieved document content and metadata (including document retriever location and similarity score) are displayed for the user.

See Figure 4-3 for a screenshot of the RAG web app interface on completion of an end-to-end FS-RAG chain invocation. The app maintains a chat history log of the user's previous queries, FS-RAG generated responses, and cited document context in similar format until the current session ends (i.e. the user logs out of the web app). The retrieved documents are displayed in descending order of relevancy score, so the topmost document (displayed in Figure 4-3) has the highest ClinicalBERT embedding cosine-similarity score of 0.898 among all EHR record documents in the federated hospital hierarchy that the logged-in user can access.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 5

Case Study

We consider a case study involving three hospital organizations, which we will address as Hospital A, B, and C. While we anticipate many details of specific hospital departmental structure to not be representative of hospitals in practice due to our lack of clinical experience, we focus our case study evaluation on the federation and strong policy/attribute-based access control aspects of the architecture, which can be flexibly adapted to accommodate different retrieval flows and/or PBAC attributes/policies.

In this case study, **Hospital A** represents a major regional hospital providing generalized care to many patients across a variety of disciplines. **Hospital B** represents a minor, local hospital serving a smaller patient community with fewer internal departments. Finally, **Hospital C** represents a highly-specialized neurosurgery clinic concentrating on medical neurology research and treatment care for patients with brain conditions that the more generalized hospitals A and B lack resources to adequately treat.

The 3 hospitals have found it mutually beneficial to share their EHR knowledge bases in a limited fashion to support question-answering on trends across all patient EHR data. They choose to implement the FS-RAG framework so that they are able to selectively control EHR records access for both internal users and external users from the two other hospitals, as well as perform federated document retrieval like a centralized system while maintaining decentralized knowledge bases. Each of the three hospitals spins up its own Flask server responsible for handling OIDC authentication and PBAC-secured federated retrieval; hospital Flask servers communicate with each other and the FS-RAG web app tool via HTTP requests (see Section 4).

5.1 Partitioning MIMIC-IV Data

Since all MIMIC-IV EHR data [15] [16] was collected from a single hospital (i.e. the Beth Israel Deaconess Medical Center), we first describe our data partitioning methodology to simulate three decentralized EHR knowledge bases for hospitals A, B, and C.

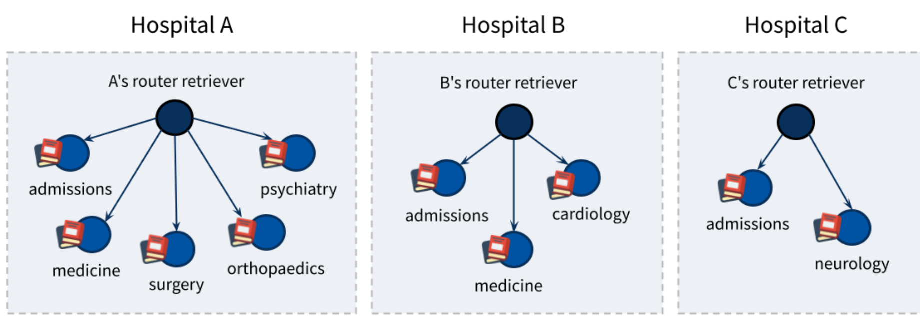

Figure 5-1: Sketch of the federated 3-hospital retrieval hierarchy in the case study. Patient EHR data is isolated to each department-level leaf retriever's knowledge base.

As sketched in Figure 5-1, our simplified case study includes 1 hospital-level router retriever for each of hospitals A, B, and C, connected to a set of leaf retrievers with ownership of patient EHR data at the department-level.

- **Hospital A**: We extracted all MIMIC-IV discharge summary notes [16] corresponding to the first 50 distinct patient `subject_id` values in the MIMIC-IV patients list (obtained from `patients.csv` in the *hosp* module) [15]. Each discharge summary note is prefixed with a `Service` field, which we use to partition the set of notes into 11 distinct clinical services. We form the unstructured EHR knowledge bases for A's leaf retrievers from the **medicine**, **surgery**, **orthopaedics**, and **psychiatry** services.

- **Hospital B**: We extracted the next 50 distinct patients by `subject_id` and associated discharge summary notes, which we filtered to the **medicine** and

cardiology services, since hospital B is characterized as a smaller hospital serving only general internal medicine and a cardiology department.

- **Hospital C**: We aggregated all discharge summaries from hospitals A and B with services labeled as **neurology** and **neurosurgery** to populate hospital C's neurology leaf retriever database. While this results in patient overlap between hospital C and each of A and B, we believe that this is not only benign but likely more realistic, as single-patient EHRs decentralized across multiple hospitals (i.e. seeking routine care at hospital A but specialized trial treatment at hospital C) are an important use case that our FS-RAG architecture intentionally supports.

In addition to unstructured discharge summary data, for each hospital, we also include a separate leaf retriever holding structured patient **admissions** EHR records corresponding to the hospital's discharge notes (obtained from `admissions.csv` in the *hosp* module [15]).

We implement all hospital retrievers with LangChain [21] (see Section 4), which requires a pre-processing step to chunk long EHR discharge summary notes into multiple documents (of max size 800 characters) that serve as the base unit for retrieval. The chunking step is necessary to ensure that final formatted prompt to the LLM, consisting of the `query`, the top k document context retrieved, and a small amount of instruction prompting, fits into the LLM context window (which is only 512 tokens for Flan-T5 [7]).

The LangChain default `CSVLoader` performs chunking by counting characters without considering the document content, so we modified it to more intelligently chunk on EHR note sentence/newline boundaries to avoid sentence fragments that may hurt document relevance and understandability. We reproduced the chunked EHR document for Janet Aguilar from Hospital A's orthopaedics leaf retriever as well as the corresponding admissions retriever entry in Table 5.1 for a concrete example of document content.

Table 5.1: Example EHR discharge summary and corresponding admissions record.

Leaf Retriever	Document Content
$A_{\text{orthopaedics}}$	For patient with name of Janet Aguilar: Service: ORTHOPAEDICS. Allergies: Codeine / Augmentin / Topamax. Chief Complaint: left knee osteoarthritis/-pain. Major Surgical or Invasive Procedure: ____: left total knee arthroplasty . History of Present Illness: ____ year old female w/left knee osteoarthritis/pain who failed conservative measures, now admitted for left total knee arthroplasty. Past Medical History: Dyslipidemia, varicose veins (R>L) s/p ligation, COPD, OSA (+CPap), recent URI (received course of Zithromax), bilateral PEs (____), antiphospholipid antibody syndrome (on lifelong anticoagulation), T2DM (last A1C 6.2 on ____, cerebral aneurysm (followed by Dr. Social History: ____ Family History: No family hx of DVT or PE, two sisters have atrial fibrillation.'
$A_{\text{admissions}}$	subject_id: 10002221 name: Janet Aguilar hadm_id: 20195471 admittime: 2203-06-13 00:00:00 dischtime: 2203-06-16 13:28:00 deathtime: nan admission_type: SURGICAL SAME DAY ADMISSION admit_provider_id: P21E8 admission_location: PHYSICIAN REFERRAL discharge_location: SKILLED NURSING FACILITY insurance: Medicare language: ENGLISH marital_status: SINGLE race: WHITE edregtime: nan edouttime: nan hospital_expire_flag: 0

5.2 Setting up Mock Users

The FS-RAG retrieval architecture must dynamically restrict the set of retrieved documents based on authorization policies on user-resource attributes to protect against EHR information leakage. We created **eight mock users holding different security properties** due to varying combinations of user authorization attribute values (i.e. hospital organization, department, role, and affiliated research groups) (see Table 5.2). Note that `A.admin` refers to administrative staff involved with handling patient admissions, insurance information, and other logistical tasks, and not "root"/superuser level access. As described in Implementation (see Section 4.2), we

Table 5.2: Table of user access control attributes for evaluation.

User	org	role	dept	affiliations
A.phys.neur	A	physician	medicine	C_neuro
A.phys	A	physician	surgery	
A.nurse	A	nurse	psychiatry	
A.tech.rad	A	technician	radiology	
A.admin	A	admin		
B.phys	B	physician	cardiology	
B.nurse	B	nurse	medicine	
C.research	C	researcher	neurology	

instantiated PBAC policy decision points from JSON `py-abac` policies at all hospital router and leaf retrievers. On retrieval for the same clinical trend query, different mock users should receive different access control allow/deny decisions at various retriever PDP points in the retrieval hierarchy. We summarize the **authorized access pattern of all eight user security profiles** over the federated 3-hospital case study in Figure 5-2, where green checkmarked boxes indicate `ALLOW` PBAC decisions and red x-ed boxes indicate `DENY` PBAC decisions for each particular PDP policy set.

userinfo	Hospital A						Hospital B				Hospital C		
	router	adm	med	psy	sur	ort	router	adm	med	car	router	adm	neu
A.phys.neur	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
A.phys	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×
A.nurse	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×
A.tech.rad	✓	×	×	×	×	✓	✓	×	×	×	✓	×	×
A.admin	✓	✓	×	×	×	×	✓	×	×	×	✓	×	×
B.phys	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	×	×	×
B.nurse	✓	✓	×	×	×	×	✓	✓	✓	✓	×	×	×
C.research	×	×	×	×	×	×	×	×	×	×	✓	✓	✓

Figure 5-2: Table of federated retriever PBAC access for all mock users (Table 5.2). Note the asymmetry between hospital policy access control schemes: hospital B grants `A.nurse` access to all router/department retrievers, but hospital A restricts `B.nurse` to only the router and admissions data.

We note that PDP `DENY` decisions at a retriever are coarse-grained and thus squash the user's query without propagating it down any retrievers in its subtree. For example, we observe that `C.research` is denied access at hospitals A and B's router-level retriever and thus is automatically unable to access any A/B leaf retrievers. On the other hand, all Hospital A users are granted access at Hospital C's router retriever despite exclusively `A.phys.neur` holding actual access to any Hospital C EHR documents at the leaf-level.

Chapter 6

Evaluation

6.1 Metrics

We evaluate our federated secure RAG pipeline along the axes of end-to-end **generated response quality** and retrieval-step **security guarantees** (i.e. preventing information leakage). We consider the following baseline RAG models:

- Centralized Insecure RAG (CI-RAG): standard question-answering RAG chain using 1 centralized retriever with an aggregated view over all decentralized hospital EHR databases. To ensure fair evaluation against the federated architectures, we construct the singular retrieval knowledge base for CI-RAG by creating a Qdrant vector store from all hospital EHR data so that the total available data for retrieval is identical.

- Federated Insecure RAG (FI-RAG): federated question-answering RAG chain without any security authentication/authorization mechanisms (see Section 3.1). Queries are propagated down and retrieved documents are aggregated up a multi-retriever hierarchy.

All CI-RAG, FI-RAG, and FS-RAG retrievers use ClinicalBERT EHR embeddings [2] and the GPT-3.5-Turbo LLM for generation [4]. **The primary difference we are interested in for evaluation lies in the *retrieval* step: specifically, from federating across multiple isolated knowledge bases (CI-RAG baseline vs**

FI-RAG) and enabling PBAC authorization filters (FI-RAG baseline vs FS-RAG). For the former, we demonstrate that, after adding *federation*, the retrieval step is able to retrieve the exact top k most relevant documents (i.e. identical to the centralized version) and thus maintain response quality in synthesizing clinical trends. For the latter, we demonstrate that adding security *access control* correctly restricts the retrieved document set based on different user security properties, and analyze the (expected) degradation in response quality due to the limited EHR context.

6.1.1 Metrics for Response Quality

We employ **ROUGE-1, ROUGE-L [23], and BLEU [25]** metrics to evaluate the generated response quality on QA tasks over clinical trends (see Section 6.2.2). When evaluating **FI-RAG against the CI-RAG baseline**, we expect that these metrics will be reasonably close to 1.0 (i.e. exactly identical response texts word/sentence/syntax-wise) to demonstrate that federating the retrieval step did not induce a loss in response quality: as the LLM generation step is identical, significantly dissimilar responses would likely result from FI-RAG retrieving a worse set of documents. We note that the GPT-3.5 generation LLM is non-deterministic (although we do set `temperature = 0` to minimize noise), so the metrics are likely to not be exactly 1.0. When evaluating **FS-RAG** under different user access profiles **against the FI-RAG baseline**, we expect that the ROUGE and BLEU response similarity metrics will be *closer to 0* for users with more *limited* access (i.e. since less highly-relevant documents are accessible) and *closer to 1* for users with *wider* access.

ROUGE [23] comprises metrics commonly used to evaluate machine translation and summarization tasks via a measurement of similarity between a reference and a target text. Intuitively, we can consider the ROUGE metrics as measures of how much of the baseline response content was also present (recalled) in the target response. ROUGE-1 calculates the percent of unigrams present in the reference response that are *also* present in the target response:

$$\text{ROUGE-1} = \frac{\sum\limits_{\text{1-gram} \in \text{ref}} \text{COUNT}_{\text{target}}(\text{1-gram})}{\sum\limits_{\text{1-gram} \in \text{ref}} \text{COUNT}_{\text{ref}}(\text{1-gram})}$$

ROUGE-L is another recall-oriented metric (i.e. computed over reference text) evaluating longest common subsequence (LCS) matches present in the reference that are also present in the target response. ROUGE-L presents a measurement of sentence-level structural similarity rather than simple unigram word overlap. Although we do include ROUGE-L scores, for our system evaluation goals, ROUGE-1 is more relevant since we are less concerned about copying sentence syntax than generating similar keywords/semantic ideas.

BLEU [25] metrics are commonly used for automatic precision evaluation on machine translation tasks. Intuitively, this captures the amount of text information in the target response (i.e. after adding federation or security control) also matching in the reference response. BLEU is computed as (under only 1 reference response):

$$\text{BLEU} = BP \cdot \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$$

with modified precision scores from the clipped counts of target n-gram matches in the reference:

$$p_n = \frac{\sum\limits_{n\text{-gram} \in \text{target}} \text{COUNT}_{\text{clip}_{\text{ref}}}(n\text{-gram})}{\sum\limits_{n\text{-gram} \in \text{target}} \text{COUNT}_{\text{target}}(n\text{-gram})}$$

where w_n are weights and BP is a brevity penalty added to avoid rewarding excessively short responses (i.e. a generated target text of “the” has perfect precision but obviously little meaning).

Finally, since ROUGE-1, ROUGE-L, and BLEU reflect exact word matching between the reference and target responses rather than a more realistic and useful similarity matching based on semantic content, we also conducted a manual inspection over the generated responses.

6.1.2 Metrics for Retrieval Security

FI-RAG does not implement authorization control, so the retrieved document set should be **identical** across all user attribute profiles – all users should retrieve the global k most relevant documents across the federated hierarchy. This potentially implies undesirable leakage of sensitive patient health information accessible to physicians but not other hospital staff, or accessible to clinicians at the local hospital but not at other hospitals, etc. In contrast, we aim for FS-RAG to retrieve **different** sets of documents depending on experimental **access privileges** dynamically granted to users based on retriever PDP policies. Since our threat model assumes security leaks occur only during the retrieval step (i.e. precluding website security attacks), these experiments help verify that each hospital's FS-RAG retriever successfully restricted retrieval to only authorized documents. To facilitate this, we propose mock users with PBAC attributes covering different cross-sections of EHR retriever/document access in a case study hospital system (see Section 5) to ensure that all responses **only cite authorized documents** even when given the **same query** and even when more query-relevant documents are present but access-forbidden. To quantitatively describe this expected retrieval information loss for each mock user u_i, we compute an intersection metric between the restricted document set $D_{fs|u_i}$ and the document set D_{fi} retrieved in the FI-RAG baseline system:

$$\text{ixn}_{u_i} = \frac{\left|D_{fi} \cap D_{fs|u_i}\right|}{|D_{fi}|}$$

While the intersection metric is not a formal verification of our system's security guarantees, we carefully select mock access profiles and clinical trend queries for high coverage of our case study's PBAC policy gates to experimentally demonstrate that ixn_{u_i} decreases or drops to 0 for FS-RAG users with limited/no access.

6.1.3 RAGAS Reference-Free Metrics

Evaluation of our system is complicated by the **lack of true "gold" documents or responses** to our ClinicalTrendQA queries, primarily due to challenges in systematically generating ground truth baselines appropriate for our targeted trend-synthesization task. While we may be able to manually annotate responses and select highly relevant documents for simpler, patient-specific queries, we believe that constructing the ground truth ourselves is highly liable to introduce personal researcher bias, especially since we have little clinical background experience. As such, it would be most suitable for licensed physicians and medical professionals to construct the ground truth responses in accordance with established medical intuition, but gathering expert medical opinion is out of the scope of this project.

Without an objective ground-truth reference for either the retrieval step (i.e. gold relevant document set) or end-to-end generation (i.e. physician-curated response), we calculate the above evaluation metrics (i.e. relevant set ixn, ROUGE-1, ROUGE-L, and BLEU) by using the FI-RAG retrieval and generation outputs as pseudo ground truth baselines against which we compare the FS-RAG outputs, as detailed previously.

We additionally consider metrics from the **RAGAS automatic RAG evaluation suite [10]** as suggested by a 2024 survey on state-of-the-art RAG models and retrieval/generation evaluation metrics [12]. RAGAS is especially suitable for our use case due to its unique ability to evaluate various dimensions of end-to-end RAG pipelines **without the need for ground-truth human annotations**. As we are particularly interested in assessing FS-RAG along the components of **retrieval** relevancy to the query and **generation** response quality, we selected the following reference-free RAGAS metrics [10]:

1. **Context Relevancy** $\in [0, 1]$ evaluates the relevancy of the retrieved document `context` content to the original `query` content. The metric is computed by chunking the retrieved `context` sentence-wise into a set $S_{context}$ of all context sentences and using an LLM to judge whether each context sentence contains

relevant information for answering the provided `query`:

$$\text{Context Relevancy} = \frac{\{s_i \in S_{context} \mid s_i \text{ is relevant to query content}\}}{|S_{context}|}$$

Higher context relevancy scores indicate better retrieved contexts, where "better" contexts are defined as **minimizing redundant** and query-irrelevant information while also **containing all essential** information relevant to the query. We consider RAGAS context relevancy for completeness in evaluating the retrieval component through assessing retrieved **content** similarity: our custom ixn_{u_i} metric evaluates the efficacy of PBAC retrieval access restrictions but only compares document IDs rather than content. However, we note that the RAGAS metric is more suitable for straightforward, fact-recall questions like "What is the capital of France?" where relevance of context sentences to the query can be easily determined. Since we target the more complex, open-ended problem of combining information from multiple documents, where each statement may only address part of the query, we expect context relevancy scores for the federated RAG pipelines to skew lower even on perfect retrieval.

2. **Faithfulness** $\in [0, 1]$ evaluates how semantically consistent the generated `response` information is with the retrieved document `context`. To compute faithfulness, RAGAS uses an LLM to identify a set of statement claims S_{claims} from the `response` text, and then asks the LLM to judge whether each claim is supported by facts in the provided `context`:

$$\text{Faithfulness} = \frac{|\{c \in S_{claims} \mid \text{c is supported by retrieved context}\}|}{|S_{claims}|}$$

   Higher faithfulness scores indicate better responses as defined by having high relevance between all response claims and document context (i.e. minimal unsupported or hallucinated claims).

3. **Answer Relevancy** (usually in $[0, 1]$) evaluates the generated `response`'s relevancy to the original `query`, where **higher scores indicate better responses**

in terms of avoiding redundant, off-topic details, or incompleteness. Importantly, the metric **does not evaluate factuality** (as it does not possess a reference gold response), but rather response content **relevancy to all parts of the query topic**. From the provided `response`, RAGAS uses an LLM to reverse-generate N (3 by default) fake queries, and calculates the mean cosine similarity of the original query q_o embedding to each response-derived fake query q_k embedding:

$$\text{Answer Relevancy} = \frac{1}{N} \sum_{1 \leq k \leq N} \cos(\mathbf{e}_{q_k}, \mathbf{e}_{q_o})$$

The query-embeddings are generated using the closed-source OpenAI general-domain **`text-embedding-ada-002`** embedding model [10].

For all three metrics, RAGAS uses the **`gpt-3.5-turbo-16k` LLM** [4] to judge relevancy between the original query, retrieved context, and response statements, or in the case of Answer Relevancy, to generate reverse-engineered queries from the response text [10]. We acknowledge that judging factuality and relevancy via computing non-LLM similarity-score metrics between the RAG context/responses and expert physician-constructed ground-truths would be preferable as it would circumvent introducing LLM bias in the evaluation framework. This is especially true since the RAGAS LLM judge and our FS-RAG generation model both use GPT-3.5-Turbo [4] (albeit one of the strongest general-domain LLMs currently).

However, the prior LLM-as-a-judge study found LLM judges like GPT-4 achieved over 80% agreement with human judge decisions across the multi-question, open-ended MT-bench benchmark and user-crowdsourced ratings of the Chatbot Arena dataset [38]. Accordingly, since obtaining physician ground truth is extremely costly and infeasible for our project scope, we believe that there is still valuable information in the evaluation metrics computed with LLM-judge frameworks like RAGAS, with a note of caution that, to our best knowledge, there has not been prior research on LLM-as-a-judge alignment with *professional clinical opinion* or other domain-specific consensus.

6.2 Creating the ClinicalTrendQA Dataset

We will source patient EHR data from the freely-available Medical Information Mart for Intensive Care (MIMIC)-IV dataset (a streamlined version of the original MIMIC-III dataset), which is a large database of deidentified EHR records from over 40,000 patients admitted to the Beth Israel Deaconess Medical Center during the period 2008-2019 [15]. In addition to structured data (i.e. physiological measurements, medications, diagnoses), MIMIC-IV health records include large quantities of unstructured text data (i.e. patient progress notes, discharge summaries), which in particular suggest it as a good candidate for assessing natural language model performance in the clinical domain.

However, MIMIC-IV tables only provide raw EHR health data; thus, evaluating response accuracy requires extending it into a question-answering task dataset from the patient medical information. Existing clinical-domain LLM research has published similar medical QA datasets, but they predominantly assess performance on single-hospital or single-patient, single-note QA tasks, such as individual clinical note inference (i.e. ClinicalBERT [2]), risk factor/diagnosis prediction (i.e. Almanac [37], zero-shot prediction for radiology [1]), or medical entrance exam knowledge recall (i.e. MultiMedQA benchmark [31]). To the best of our knowledge, there are no existing dataset benchmarks for the specific question-answering task of clinical *trend recognition* to test extraction of information laterally across multiple patient EHRs.

Thus, we construct **ClinicalTrendQA** (see Section 6.2.2), a dataset of 20 (question, answer) pairs informed by the MIMIC-IV patient EHR database [15]. The questions will involve identifying and synthesizing **lateral clinical trends** necessitating retrieval of information originating from **multiple knowledge bases across federated organizations and access control boundaries**.

6.2.1 MIMIC-IV Overview

The MIMIC-IV dataset [15] is divided into two major modules, both providing **structured** patient EHR data (i.e. lab test results, prescriptions, demographic information): a general hospital-wide database store (*hosp*) and an ICU-specific database store (*icu*) [15]. An additional module *note* contains unstructured EHR data, such as patient discharge summaries and radiology imaging reports [16]. Since our goal for the federated secure RAG pipeline is to synthesize trends across EHR records, it is important to support federated retrieval over information contained in both structured and unstructured clinical data. While our architecture is agnostic to the specific module or type of EHR data utilized, we chose to demonstrate FS-RAG retrieval over unstructured patient **discharge summary** text (from `discharge.csv` in the *note* module) and structured hospital **admissions** records (from `admissions.csv` in the *hosp* module). We elected to source unstructured data from discharge summaries rather than the more specialized radiology reports since we aim to evaluate RAG responses across a general population of patients and medical conditions. In addition, while FS-RAG does not impose restrictions on the type of structured EHR data, we observed that many structured MIMIC-IV tables (i.e. `labevents`, `diagnoses`) are redundant with the unstructured physician-written discharge summaries, so we selected the admissions table since it provides unique demographic information.

6.2.2 ClinicalTrendQA

The ClinicalTrendQA dataset, reproduced in Table 6.1, emphasizes performance on three flavors of EHR trend-retrieval:

- **Single-patient, single-note retrieval (2 queries)**. This is similar to the task explored by Ahsan et. al. [1], in which the LLM performs inference over 1 clinical note; however, the FS-RAG pipeline, rather than the physician, is expected to retrieve the correct EHR note among all accessible knowledge bases to provide to the generation step.

- **Single-patient, multiple-note retrieval (9 queries)**. FS-RAG must also be able to extract information from multiple notes, which may be arbitrarily located in any set of hospital/department knowledge bases. This reflects the realistic scenario where a patient visits and thus has EHR information distributed across multiple clinics: for example, they could see a routine family doctor but consult outside specialists for mental health, eye health, and other medical concerns. FS-RAG should retrieve the patient's entire medical history (assuming authorized access) to best inform its responses and the clinical decisions they potentially advise.

- **Multiple-patient retrieval (9 queries)**. We also evaluate FS-RAG on broader queries regarding clinical trends across the general patient population. Ideal responses should, as much as possible under the constraints of the user's authorized access, retrieve from multiple patient EHR documents to compose a balanced response without extrapolating one patient's data to the general population.

Table 6.1: Table of the ClinicalTrendQA dataset annotated by trend QA task: single-patient/single-note (boxed), single-patient/multiple-note, and multiple-patient (circled).

#	**ClinicalTrendQA Query**
1	For patients admitted to this practice who report heavy alcohol consumption, what are some of the most common diagnoses?
2	For patients receiving general surgery, what are the most common chief complaints?
3	What are some complications experienced by patients who underwent coronary artery bypass?
4	What are some common diagnoses for patients reporting abdominal pain?
$\boxed{5}$	Has the prescribed medication impacted Edward Fisher's behavior, such as reported incidents of outbursts or other impulsive actions?
6	Has Terry Ruschmeyer been compliant with taking her prescribed medications?
7	What procedures has Carolann Bartholf received for complications following her craniotomy?
8	What have been the prior causes of abdominal pain for patient Alice Johnson?
9	Did Alexandra Edwards experience anemia or low blood levels, and what were the causes?
10	For patient Janet Aguilar, how did her breasts heal after her breast lumpectomy?
11	What have been Betty Henry's experiences with auditory hallucinations?
12	How have Harriet Gutierrez's complaints of left-sided symptoms progressed over time?
13	Describe the progression of left-sided weakness for Tracy Thompson.
14	What are the most significant risk factors for stroke among neurology patients?
15	What cardiothoracic surgical procedures has Lee Hiatt undergone during his hospital admissions?
$\boxed{16}$	What were the findings from David Carvalho's barium swallow during his esophageal study?
17	Is Medicare or Medicaid more widely used for insurance among patients at Hospital A?
18	For patients admitted for cardiology or cardiothoracic services, what was the most common surgical procedure performed?
19	What are the most severe withdrawal symptoms experienced by patients with a history of drug or alcohol use, and how were withdrawal symptoms treated?
20	Is it more common for patients to present to the emergency department for orthopedic surgery due to a mechanical fall, or due to pain from preexisting health conditions?

6.3 Results and Discussion

6.3.1 IXN Scores (verifying retrieval authorization)

To evaluate the correctness of our PBAC authorization mechanisms in restricting retrieval-step document access, we performed experiments with a varied array of mock user security properties as detailed in our case study setup (see Figure 5-2). As described at the start of the Evaluation section, we computed an **$\mathbf{ixn}_{u_i}$ score** for each user u_i by calculating the set intersection between document IDs retrieved by u_i under FS-RAG retrieval and reference document IDs retrieved by FI-RAG for the same query. All experiments for the intersection metric used the **hyperparameter** $k = 10$, or **top-10** most relevant documents returned after the retrieval step. Since FS-RAG retrieval differs from FI-RAG retrieval only in its additional policy-based access control checks, scores with $\text{ixn}_{u_i} < 1$ imply that the user was denied access to a subset of the most relevant documents. Specifically, lower scores imply more restrictive access privileges (i.e. less access) and higher scores imply more permissive access privileges (i.e. more access). We can cross-reference our ixn metric results with the true access privilege properties defined by our case study policies in Figure 5-2 to confirm that the intersection scores match each user's granted access profile.

The **resulting $\mathbf{ixn}_{u_i}$ scores are summarized in Table 6.2** (note: FI-RAG is abbreviated to "fi" for space). We also include the average ixn_{u_i} score across all 20 ClinicalTrendQA questions for a coarse indication of the *power* of each user role (i.e. where more power suggests a user with more permissive access properties). We observe that the federated insecure retrieved document sets, when compared against those retrieved by the centralized system, are all **1.0** across all questions, which indicates that FI-RAG and CI-RAG consistently retrieved identical documents. This matches our expectations since splitting retrieval across a federation of knowledge bases should not change the *set* of most relevant documents ultimately retrieved, only the *method* of retrieval.

Table 6.2: Retrieved document set intersections between all scenarios and their baselines. The federated **insecure** RAG (**fi**) scenario was compared against centralized insecure (ci) RAG as a baseline. All federated **secure** RAG user-profile scenarios were compared against fi as a baseline.

Q#	**fi**	**A. phys**	**A. phys. neur**	**A. tech. rad**	**A. ad-min**	**A. nurse**	**B. phys**	**B. nurse**	**C. re-search**
1	1	1	1	0	0	1	1	0.5	0
2	1	0.8	1	0	0	0.8	0.8	0.4	0.2
3	1	1	1	0	0	1	1	0.5	0
4	1	1	1	0	0	1	1	0.3	0
5	1	1	1	0	0	1	1	0	0
6	1	1	1	0	0	1	1	0	0
7	1	0	1	0	0	0	0	0	1
8	1	1	1	0	0	1	1	0	0
9	1	1	1	0	0	1	1	0	0
10	1	1	1	0.4	0	1	1	0	0
11	1	1	1	0	0	1	1	0	0
12	1	0	1	0	0	0	0	0	1
13	1	0.5	1	0	0	0.5	0.5	0	0.5
14	1	0.9	1	0	0	0.9	0.9	0.1	0.1
15	1	1	1	0	0	1	1	1	0
16	1	1	1	0	0	1	1	1	0
17	1	0.9	1	0.1	0	0.9	0.9	0.1	0.1
18	1	1	1	0	0	1	1	0.5	0
19	1	0.9	1	0	0	0.9	0.9	0.3	0.1
20	1	0.9	1	0	0	0.9	0.9	0.3	0.1
Avg.	1	0.845	1	0.025	0	0.845	0.845	0.25	0.155

We now consider our experimental retrieved document set intersections for various users with the FS-RAG paradigm against the baseline FI-RAG retrieved sets. The user **A.phys.neur** is granted functionally ubiquitous access in the evaluated system (see Figure 5-2) since all retriever PDPs have at least one policy authorizing access to A.phys.neur. As we expect, $\text{ixn}_{\text{A.phys.neur}} = 1.0$ across all queries, indicating that FS-RAG correctly retrieved the set of most relevant documents identical to the insecure baselines when an all-access user is authenticated in the system. In contrast, the high (but not all) access **A.phys** user is granted access to all EHR documents in the system except those located in Hospital C's knowledge bases (i.e. $C_{\text{admissions}}$ and $C_{\text{neurology}}$). We accordingly observe $\text{ixn}_{\text{A.phys}}$ scores of 1 or close to 1 for the majority of questions. Some notable exceptions include $\text{ixn}_{\text{A.phys}} = 0.5$ for **question 13** about Tracy Thompson (discussed in-depth further down), and $\text{ixn}_{\text{A.phys}} = 0$ (no access to any baseline-retrieved relevant documents) for questions **7** ("What procedures has Carolann Bartholf...craniotomy?") and **12** ("How have Harriet Gutierrez's complaints of left-sided symptoms...?"). Further analysis of the specific documents retrieved by the all-access FI-RAG baseline on those questions revealed that a significant subset of the baseline retrieved set were located at the $C_{\text{neurology}}$ retriever: 50% (question 13), 100% (question 7), and 100% (question 12). Since $C_{\text{neurology}}$ should deny access to A.phys, **this relevant document distribution exactly matches the experimentally observed $\text{ixn}_{\text{A.phys}}$ pattern**.

Limited access users are **A.admin**, which only has access to structured admissions records in Hospital A, and **A.tech.rad**, which only has access to orthopaedic documents in Hospital A (see Figure 5-2). We found that $\text{ixn}_{\text{A.tech.rad}} = 0$ for almost all queries, but notably $\text{ixn}_{\text{A.tech.rad}} = 0.4$ for question 10 ("For patient Janet Aguilar...breast lumpectomy?"). Further examination revealed that of the top 10 most relevant documents under all access, 4 were located in A's orthopaedics retriever as expected.

We also found that $\text{ixn}_{\text{A.admin}} = 0$ for all questions, which indicates that *none* of the top 10 relevant documents accessible to A.admin were in the top 10 *globally* most relevant documents across all knowledge bases. This makes sense since the clinical trend queries target medical information found in discharge summary notes rather than the demographic data in admissions records. The only exception is question 17 ("Is Medicare or Medicaid more widely used...at Hospital A?"), which requires insurance information from admissions documents and thus should have resulted in $\text{ixn}_{\text{A.admin}} = 1$. The fact that the all-access baseline determined that discharge summary documents, without any relevant insurance information, were more relevant than the structured admissions entries suggests that our ClinicalBERT [2] embed-

Table 6.3: Average document intersection, ROUGE, and BLEU scores between all scenarios and their baselines.

Scenario	Baseline	$\mu_{\mathbf{ixn}}$	ROUGE-1	ROUGE-L	BLEU
fi	ci	1	0.78	0.71	0.63
A.phys.neur	fi	1	**0.80**	**0.75**	**0.65**
A.phys	fi	0.85	0.65	0.58	0.47
A.nurse	fi	0.85	0.63	0.55	0.43
A.tech.rad	fi	0.03	0.40	0.28	0.16
A.admin	fi	0	*0.35*	*0.25*	*0.10*
B.phys	fi	0.85	0.66	0.57	0.47
B.nurse	fi	0.25	0.44	0.36	0.22
C.research	fi	0.16	0.45	0.35	0.20

dings may not be suitable for determining relevance with structured data, but further research is necessary to draw a firm conclusion.

We demonstrated that our FS-RAG architecture sufficiently performs authorization-enhanced retrieval that dynamically serves different documents to users with different security properties. The computed ixn scores verify that users with more restrictive access privileges retrieved smaller subsets of the most relevant documents, and further inspection of the exact document IDs confirms that we successfully restricted user access to **prevent any information leakage** (*no* documents retrieved were unauthorized) while also guaranteeing that **all accessible relevant documents were retrieved** (correctness *after* satisfying the prior constraint) for the 20 clinical trend queries and 8 user security profiles evaluated by our case study experiments.

6.3.2 ROUGE & BLEU (end-to-end responses)

The average ROUGE-1, ROUGE-L, and BLEU scores (defined in Section 6.1.1) across the 20 ClinicalTrendQA tasks for all user scenarios are summarized in Table 6.3, with the highest and lowest score of each metric column highlighted as **bold** and *italic* text, respectively. We observed that the all-access **A.phys.neur user consistently achieved the highest scores** (ROUGE-1=0.80, ROUGE-L=0.75, BLEU=0.65, which are all close to 1) among all 8 experimented user security profiles. This is reasonable since it has access to documents at all knowledge bases and thus its generated responses are informed by the same sets of most-relevant information as the baseline insecure responses. We note that A.phys.neur's responses achieved slightly better scores against the FI-RAG responses than the FI-RAG responses achieved against the CI-RAG ones, but the difference is small (0.02 for ROUGE-1 and BLEU and 0.04 for ROUGE-L) and can likely be attributed to variability inherent to GPT-3.5 generation LLM since the all-access A.phys.neur, FI-RAG, and CI-RAG scenarios retrieve the same document context and thus should only differ due to generation non-determinism. We also found that **A.admin consistently achieved the worst scores across all generation metrics** (ROUGE-1=0.35, ROUGE-L=0.25, BLEU=0.10), which matches our expectations since A.admin had no access to the most query-relevant documents for almost all queries (i.e. $\mu_{\text{ixn}_{\text{A.admin}}} = 0$) and thus should not be able to compose correct responses.

In interpreting response quality from the ROUGE and BLEU scores, we note that these metrics are computed from exact word matching rather than semantic content, the latter of which would be more meaningful for our evaluation goals since we are less concerned with response syntax than the synthesized information. Since we verified that the FS-RAG retrieval step correctly collected the most relevant documents under each user's access constraints via intersection scores and manual inspection of document IDs, we believe LLM non-determinism caused ROUGE/BLEU scores to skew lower (for example, less than 1 for A.phys.neur). However, comparison of ROUGE/BLEU scores across users is still interesting as it reveals that generated response quality (measured by comparing with the insecure baseline) degrades as access privileges become more restrictive, which is sensible due to the decreasing number of relevant documents. This is also likely why our observed ROUGE-L scores, which penalizes deviations in sentence structure more, are consistently lower than the ROUGE-1 scores for the same user access privileges.

To better understand the impact of retrieval-step access *restrictions* on end-to-end response quality *degradation*, we plotted **ROUGE-1 vs ixn** for the 5 users

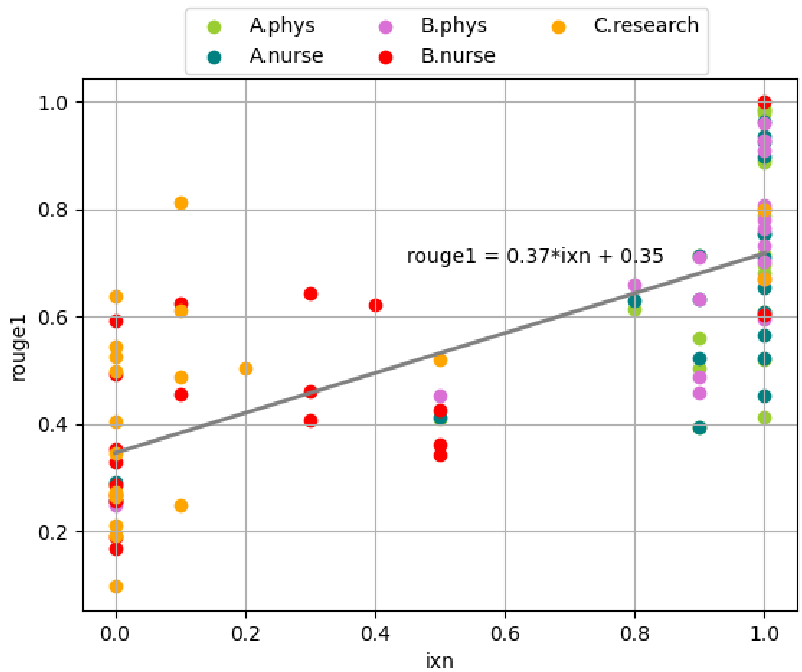

Figure 6-1: Plot of **ROUGE-1** vs retrieval-set **ixn** scores for 5 selected users with highest variability in relevant document access across all ClinicalTrendQA queries (i.e. highest standard deviations σ_{ixn}).

(A.phys, B.phys, C.research, A.nurse, and B.nurse) with the **highest standard deviation in intersection scores** $\sigma_{\text{ixn}_{u_i}}$ in Figure 6-1. We omitted users with all-access (A.phys.neur) and consistently limited access across all questions (A.tech.rad, A.admin) since the trend of response ROUGE-1 scores across the 20 queries is more insightful for users with higher variance in retrieval access across the queries.

Each point on the plot represents the retrieved document intersection score and generation ROUGE-1 score for one particular user and query. For example, the red point at approximately (ixn=0.3, rouge1=0.4) indicates that B.nurse, on that particular query, retrieved 3 out of the 10 most relevant documents. A fairly low ixn score like 0.3 implies that B.nurse is likely missing significant pieces of information relevant to answering the query, which is corroborated by the lower ROUGE-1 score of 0.4. We also plotted a linear regression through the scatterplot that has a positive slope confirming that increasingly restrictive document access during retrieval is correlated with declining quality in final response answers. However, the plot is fairly noisy, especially for extremely permissive (ixn=1) or extremely restrictive (ixn=0) scenarios. Interestingly, some responses from C.research (orange) under very limited access achieved surprisingly high ROUGE-1 scores, especially the point at roughly (ixn=0.1, rouge1=0.8). This may be due to realistic hallucinations in the generated response potentially inflating the ROUGE-1 score by including many similar medical keywords to the query topic even if the content is completely fabricated.

6.3.3 RAGAS Scores (context and response relevancy)

We additionally computed the Context Relevancy, Faithfulness, and Answer Relevancy RAGAS metrics (defined in Section 6.1.3) from the ClinicalTrendQA queries and our experimental retrieved contexts and responses. The RAGAS scores across all user scenarios are included in **Table 6.4**. The highest (best) and lowest (worst) scores of each metric column are highlighted in **bold** and *italic* text, respectively.

Table 6.4: Average RAGAS Context Relevancy, Faithfulness, and Answer Relevancy scores between all scenarios and their baselines.

Scenario	Context Relevancy (↑)	Faithfulness (↑)	Answer Relevancy (↑)
fi	0.026	**0.881**	**0.872**
A.phys.neur	0.026	0.864	**0.872**
A.phys	0.029	*0.737*	0.778
A.nurse	0.029	0.829	0.776
A.tech.rad	0.002	0.747	0.475
A.admin	*0.000*	0.808	*0.344*
B.phys	0.030	0.833	0.727
B.nurse	0.008	0.790	0.483
C.research	**0.037**	0.797	0.624

Faithfulness (measuring the factual consistency of the response with the retrieved context) and **Answer Relevancy** (measuring the relevancy of the response statements to answering all parts of the query) are **generation** metrics. Since our authorization security enhancements exclusively modify the retrieval step model (to obtain the context), faithfulness scores primarily indicate the quality of the GPT-3.5-Turbo LLM [4] we employ to generate the evaluated responses. Accordingly, we observe **high faithfulness scores** (all in the range [0.737, 0.881], which is considerably good as faithfulness is 1.0 at maximum) across all scenarios, which suggests that the generation LLM has good ability to compose statements supported by evidence from the contextual documents. **Answer Relevancy is highest at 0.872 for the all-access federated insecure and A.phys.neur scenarios**, which is also reasonable since they have complete access to all information relevant to answering the query. We found that **A.admin** had substantially the **worst answer relevancy score (0.344)**, likely since its restricted document access resulted in very little relevant information to the query and thus the majority of its responses simply declined to answer due to "insufficient information". Although such non-responses explain the low answer relevancy score, this is ideal behavior of our system for limited-access users without access to relevant document context.

Context Relevancy is a **retrieval** metric that examines how many **sentence-granularity** statements in the retrieved context are relevant to answering the query. As we hypothesized in Section 6.1.3, we saw very poor context relevancy scores across all user scenarios, including FI-RAG and A.phys.neur even though they had access to all relevant documents across all ClinicalTrendQA queries. To reconcile the conflict between these deflated context relevancy scores and our document ID intersection scores in Table 6.2, we consider the context relevancy calculation used by the RAGAS framework. RAGAS context relevancy appears to target simple fact-recall queries like "What is the population of New York in 2020?" or "Can accounts of type `[TYPE]` rent vehicles overnight?" for a domain-specific example. For this class of query-answer tasks, RAGAS's choice of sentence-level granularity may be appropriate since the query can be answered using information from a few sentences (i.e. "The population of New York in 2020 was `[---]`" or "No, accounts of type `[TYPE]` can only rent cars between the hours 11:00 - 17:00."). In contrast, our clinical trend queries are much broader, even if they specify a patient name, and multiple correct answers can conceivably exist for the multiple-patient queries. Thus, it is unrealistic for one sentence or even one document to contain all information relevant to the question; rather, such a 1-document inference task is not the focus of our system. As a result, each retrieved sentence likely only contributes a small amount of information to answering the query: our system endeavors to combine all of these information fragments to infer broad clinical trends. Thus, although we discuss context relevancy scores for completeness, we believe the metric's sentence-granularity premise is too coarse to be sufficient in evaluating our use case in retrieving multi-document distributed context.

6.3.4 Manual Inspection

In consideration of the limitations of the automatic ROUGE, BLEU, and RAGAS generation metrics, we also assess response quality through manual inspection.

Responses under Varying Access

In this section, we present an in-depth walkthrough of FS-RAG performance for a single clinical trend query under varying user security characteristics (see Section 6.3.1 for quantitative evaluation over all ClinicalTrendQA tasks). We selected the query **"Describe the progression of left-sided weakness for Tracy Thompson."** (Q13 in Table 6.1): since Tracy Thompson has EHR discharge summary notes located at both Hospital A's medicine retriever and Hospital C's neurology retriever (see Table 6.5), the query will trigger different retrieved documents depending on access to the two relevant leaf retrievers.

The bolded EHR `note_id`s in Table 6.5 contain documents (after chunking) that were retrieved by the centralized/federated insecure RAG pipelines. Since both baselines transparently grant access to all EHR data, we consider these EHR documents *pseudo-gold* documents (as we do not have the ability to determine and justify gold documents with physician input) that contain information *most relevant* to the query.

Table 6.5: Distribution of all Tracy Thompson's EHR notes.

A_{medicine}	10003299-DS-5, **10003299-DS-8**, **10003299-DS-9**, **10003299-DS-10**
$C_{\text{neurology}}$	10003299-DS-4, **10003299-DS-6**, **10003299-DS-7**

We note that although more *raw* EHR notes from A_{medicine} contained relevant information, after pre-processing the notes and chunking, the insecure baseline systems retrieve 5 *documents* from A_{medicine} and 5 *documents* from $C_{\text{neurology}}$. Thus, we expect users with access to only one retriever of the two will retrieve 50% of the pseudo-gold relevant document set (and 50% less relevant documents). We confirm this behavior in the table of FS-RAG vs FI-RAG document set intersections (see Table 6.2): users A.phys, A.nurse, B.phys have $\text{ixn}_{u_i} = 0.5$ (i.e. retrieved half of the relevant documents) since they have access to A's medicine but not C's neurology retriever, and user C.research also has $\text{ixn}_{\text{C.research}} = 0.5$ for the opposite reason. We note that the only FS-RAG user profile retrieving all relevant documents is A.phys.neur ($\text{ixn}_{\text{A.phys.neur}} = 1.0$), which makes sense since they are the only user with access to both A_{medicine} and $C_{\text{neurology}}$.

We selected the users A.phys.neur, A.phys, C.research, and B.nurse to analyze since they collectively cover all cross sections of possible access to the A_{medicine} and $C_{\text{neurology}}$ leaf retrievers. For each user, we transcribe their access security properties and end-to-end FS-RAG generated response, abbreviated to just the relevant sections for space, into Table 6.6. We annotate the responses with **bolded, green** text and *italicized, purple* text to mark the distribution of information cited in the end-to-end responses originating from documents in A_{medicine} and $C_{\text{neurology}}$, respectively.

From Table 6.6, we observe that the response for A.phys, with only access to A's leaf retrievers, has knowledge of Tracy Thompson's confusion resolving with **maintenance fluids** and mentions that her left-sided weakness may have worsened due to **dehydration, deconditioning, and malnutrition**, which is missing from the C.research response. On the other hand, the response for C.research, with only access to C's leaf retrievers, cites her left **SCM** weakness, left-side **brisk reflexes**, symptoms worsening over specifically a **4-week** period, and **intact sensation/proprioception**, among other $C_{\text{neurology}}$-exclusive information that is notably absent from A.phys's response. Additionally, the response for A.phys.neur synthesizes information from **both** the A and C retrievers, citing her **SCM/deltoid/other left arm weakness, brisk reflexes on the left side, 4-week period timeframe, and intact sensation/proprioception** from $C_{\text{neurology}}$ and mentioning that her neuro-

Table 6.6: Selected annotated responses for users with various access privileges on the query: "Describe the progression of left-sided weakness for Tracy Thompson."

User	**Access to** $A_{\textbf{medicine}}$	**Access to** $C_{\textbf{neurology}}$	**Annotated Response**
A.phys.neur	YES	YES	The progression of left-sided weakness...started with acute onset left leg weakness that evolved to involve the *left arm*...Her exam on admission showed weakness in the *left SCM, deltoid, triceps, and mild weakness in the left IP, as well as brisk reflexes on the left side.* The neurologic symptoms were believed to be a **recrudescence of prior stroke symptoms due to dehydration, deconditioning, and malnutrition.**...she had a history of *multiple strokes in the right posterior MCA/PCA watershed region.* The notes also mention that her left-sided weakness progressively worsened over a *4-week period*, leading to falls, but she had *intact sensation and proprioception*.
A.phys	YES	NO	The progression of left-sided weakness for Tracy Thompson is described as **subacute worsening of her chronic left-sided hemiparesis and confusion**...**her confusion resolved with maintenance fluids**...This suggests that Tracy's left-sided weakness has been a chronic issue that worsened subacutely, possibly due to factors such as **dehydration, deconditioning, and malnutrition.**
C.research	NO	YES	Initially, she was noted to have *left SCM weakness, mild L pronation with upward drift, and mild left deltoid strength.* She also had *brisk reflexes at left biceps* and patella...In a subsequent hospital admission, Tracy had progressively *worsening left leg weakness and falls over a 4-week period. She had no signs of neuropathy, intact sensation, and intact proprioception.* Overall, Tracy's left-sided weakness has evolved from mild weakness in specific muscle groups to more generalized weakness and functional impairment over time...
B.nurse	NO	NO	[T]here is no specific mention of the progression of left-sided weakness for Tracy Thompson. Therefore, there is insufficient information to answer this question accurately.

logic symptoms may have originated due to **dehydration, deconditioning, and malnutrition** from A_{medicine}. This matches our expectations since A.phys.neur has authorized access to all relevant documents (as determined by comparing to the insecure CI-RAG/FI-RAG baseline documents) to Tracy Thompson's left-sided weakness. Finally, we note that the response for B.nurse, who does not have access to either relevant leaf retriever and thus does not have any information on Tracy Thompson, acknowledges that there is **"insufficient information to answer this question accurately"** rather than fabricating a response, which is the ideal behavior for our system for users with limited access (we discuss instances of FS-RAG hallucinations and limitations in the following section).

In conclusion, we examined the responses of our FS-RAG-framework case study implementation on a query regarding a specific patient (Tracy Thompson) whose relevant medical information is federated across two hospital knowledge bases, each protected independently by retriever PBAC policies. The responses collectively verify that EHR document information from A's medicine and C's neurology retrievers is only revealed to users holding authorized access to the respective hospital retriever, demonstrating FS-RAG's ability to protect sensitive data leakage at the retrieval step. If we alternatively used the insecure FI-RAG architecture instead, even though federating retrieval addresses the privacy issues of centralizing identifiable patient EHR records, its lack of document authorization mechanisms causes the FI-RAG response to undesirably expose all of Tracy Thompson's information from both hospitals (i.e. similarly to the all-access A.phys.neur) for all users regardless of attributes.

We briefly discuss a similar example but with a different query using the selected responses in **Table 6.7**. The responses targeted the same query (question 8) for users with complete access (A.phys.neur) and no access (B.nurse) to the query-relevant documents. Likewise to the earlier discussed responses in Table 6.6, we observed that the all-access A.phys.neur response successfully answers the query (highlighted in **bold**) with supporting information extracted from multiple documents. The no-access B.nurse response is not able to retrieve any of the most relevant documents (in this case, this is because Alice Johnson's EHRs are all located in Hospital A), and thus the response correctly judges lack of information (highlighted in **bold**). Thus, the FS-RAG pipeline adequately performed dynamic retrieval such that users with access are still served high-utility responses but sensitive EHR information, even if essential to the query, is not exposed to users without access.

Table 6.7: Comparison of selected responses between users with and without access to relevant docs for the query: "What have been the prior causes of abdominal pain for patient Alice Johnson?"

User	Doc Access	Response
A.phys.neur	YES	The prior causes of abdominal pain for patient Alice Johnson have been **recurrent diverticulitis**. She has a history of sigmoid diverticulitis and has undergone a laparoscopic sigmoid colectomy for this condition. Additionally, she has experienced **RLQ abdominal pain** associated with diverticulitis flare-ups.
B.nurse	NO	Based on the provided clinical notes, there is **insufficient information** to determine the prior causes of abdominal pain for patient Alice Johnson. The **notes do not mention** any specific causes of abdominal pain that she has experienced in the past.

Hallucinations and Limitations

By encapsulating authorization in the prior retrieval step, FS-RAG ensures that LLM generation will never leak unauthorized EHR document data since no authorized documents are exposed to the generation stage in the first place. However, our system reduces but is **still vulnerable to LLM issues with hallucinating responses** when a user has limited access to true relevant documents. We reproduced two instances of hallucinated FS-RAG responses in Table 6.8, with the primary hallucinated content annotated in **bold**.

The first response is a straightforward example of a traditional LLM hallucination. A.tech.rad does not have access to any of Tracy Thompson's EHR documents necessary to answer the query, so the response should ideally admit "insufficient information". However, we observed that the response contains completely fabricated claims like Tracy requiring a wheelchair and finding difficulty with buttoning her shirt, which is not written in any of her actual EHR records. The second hallucination is caused by the LLM assuming that the query is applicable to all document context even when not explicitly stated. For example, while it is true that Nancy Jones and Carol Sellers were admitted to the emergency room/observation as evidenced from their admissions records, none of the admissions documents mention medical details like orthopedic issues following falls, which we conclude resulted from the generation LLM erroneously merging the query-topic (orthopaedic issues and mechanical falls) with its retrieved context. We categorize both issues as GPT-3.5 LLM hallucinations rather than retrieval security-related problems due to no leakage of legitimate private patient data. Future work on the FS-RAG pipeline could consider further prompt-engineering the LLM to discourage such hallucinations, or hard-coding an early return check right after retrieval to bypass the generation step entirely if the retrieved document relevancy scores are below a threshold.

Although retrieval-augmented generation alleviates LLM hallucinations by providing domain and patient-specific context and PBAC security prevents divulging unauthorized data at retrieval, our RAG system still suffers to a degree from LLM hallucinations. However, improvements in LLM models can be easily integrated into our pipeline by swapping out the generation step, and we cite all retrieved documents as supporting evidence along with the generated response. We encourage medical professionals to always manually validate FS-RAG responses against the retrieved documents (which always contain correct information as they are extracted directly from EHR records) to minimize the impact of undetected hallucinations. In addition, there is future potential in developing interpretability metrics from the cited document context and response that can provide insight into the trustworthiness of the response in terms of being supported by the document content.

Table 6.8: Select responses showing instances of LLM hallucination.

User	Q#	Hallucinated Response
A.tech.rad	13	Based on the clinical notes provided, Tracy Thompson's left-sided weakness has been progressively worsening over the past week. Initially, **she was able to walk with a cane, but now she requires a wheelchair for mobility**. Additionally, she has difficulty with fine motor tasks on her left side, such as **buttoning her shirt**. This suggests a decline in motor function and coordination on her left side. It is important for healthcare providers to monitor Tracy's left-sided weakness closely to assess for any further progression or potential underlying causes.
A.admin	20	Based on the provided clinical notes, it is **more common for patients to present to the emergency department for orthopedic surgery due to a mechanical fall rather than pain from preexisting health conditions**. This is evident from the fact that several patients, such as Nancy Jones and Carol Sellers, were admitted to the emergency room for **orthopedic issues following falls**, as indicated by their admission types and locations. For example, Nancy Jones was admitted multiple times to the emergency room for orthopedic issues **related to falls**, and Carol Sellers was admitted for observation **after a fall** as well. Therefore, based on the information provided in the clinical notes, it appears that mechanical falls are a more common reason for patients to present to the emergency department for orthopedic surgery compared to pain from preexisting health conditions.

6.4 Examining Response Quality Across LLMs

We selected the GPT-3.5-Turbo [4], Flan-T5-Large [7], and Meditron-7B [5] LLMs as candidates for generation. We first provide a brief overview of characteristics for each LLM:

- **GPT-3.5-Turbo** [4] is a popular closed, **general**-domain LLM with a context-length of **16K tokens** and billions of parameters (exact number unknown).
- **Flan-T5-Large** [7] is an instruction-finetuned **open**-source, **clinical domain**-specific LLM with a very short context-length of **512 tokens** and 780M parameters.
- **Meditron-7B** [5] is an **open**-source, **clinical domain**-specific LLM with a

context-length of **2K tokens** and 6.74B parameters.

We had a number of concerns in determining the choice of LLM to use for generation beyond response quality. In particular, due to the privacy-sensitive premise of the EHR data our system handles, using a local LLM for generation is preferable (and potentially necessary on deployment to comply with HIPAA laws) to a hosted, closed LLM like GPT-3.5. Since Flan-T5-Large and Meditron-7B are both finetuned to clinical-domain data, their generated responses could potentially contain more helpful medical knowledge than general-domain LLMs for our clinical domain-specific queries. On the other hand, one of the primary motivations for retrieval-augmented systems is to circumvent the need for expensive fine-tuning of general-domain LLMs to handle domain-specific data. Thus, we believe it is important to examine the responses of the non-clinical domain GPT-3.5 LLM when provided retrieval document context for more comprehensive insight on FS-RAG performance under LLMs with different prior knowledge. Finally, we note that Flan-T5-Large has a prohibitively short context-length of 512 tokens that is challenging to accommodate for the retrieval-generation task we are interested in since the retrieved EHR documents can make the final context-query prompt very long yet are essential to generating relevant responses.

We now present a qualitative analysis of the LLM responses in **Table 6.9**. For fair evaluation of the generation component in isolation, we remove variability that may be introduced in the retrieval step such that all LLMs receive the same query and most relevant documents formatted with the same prompt template instructions. Since the Flan-T5-Large context window of 512 token was a bottleneck among the three LLMs, we reduced the number of retrieved documents to $k = 4$ (instead of $k = 10$ for our prior experiments) and further reduced the maximum chunked document length to 400 characters (from 800 previously) to ensure that the final context-query combined prompts can fit into all three LLM context windows.

The selected results in Table 6.9 are representative of the responses generated for all 20 clinical trend queries. We observed that GPT-3.5-Turbo and Flan-T5-Large were both able to generate generally on-topic responses; for our selected query about heavy alcohol consumption, for example, GPT-3.5-Turbo mentions patients with

Table 6.9: Comparison between LLM responses on the query "For patients admitted to this practice who report heavy alcohol consumption, what are some of the most common diagnoses?"

LLM	Response
GPT-3.5-Turbo	For patients admitted to this practice who report heavy alcohol consumption, some of the most common diagnoses may include elevated transaminases in a classic 2:1 pattern for EtOH for patients like Betty Henry, as well as substance abuse/dependence for patients like Betty Henry who have a history of heavy alcohol and crack cocaine use. Additionally, patients may also present with psychiatric issues such as panic attacks or psychosis, as seen in patients like Edward Fisher and Cory Goodman. It is important to assess and address both the physical and mental health issues that may arise in patients with a history of heavy alcohol consumption.
Flan-T5-Large	<pad>Substance abuse/dependence: Patient reports a history of heavy alcohol and crack cocaine use for which she has undergone multiple detoxes. However, she reports only occasional use in recent months/years. While acknowledging that both substances worsened her mood, she denied any other withdrawal symptoms and
Meditron-7B	### Expected output ``` ### Expected output ``` ...(repeated LLM output truncated for brevity.)

"substance abuse/dependence...heavy alcohol and crack cocaine use" and Flan-T5-Large directly quotes a section of the same patient's EHR document concerned with her alcohol and substance abuse. However, we found that GPT-3.5-Turbo showed consistently better responses in terms of **integrating information** from multiple documents and **composing a logical conclusion answering the query** from the synthesization of contextual data. On the other hand, Flan-T5-Large struggled with answering the query: instead, as seen in Table 6.9, its responses strictly cite a portion of one EHR document without explaining the document's relevance to the query. In addition, Flan-T5-Large was not able to combine citations from multiple document sources even when information necessary to answer the query was located across multiple documents, rendering it insufficient for our task of reasoning trends from multi-document data.

We were not able to generate coherent responses with Meditron-7B on any of the experimented queries. The responses varied between continuously repeating the query statement until the maximum output length was reached and blank responses like the one included in Table 6.9. Given Meditron-7B's claim to performance comparable to state-of-the-art closed LLMs like Med-PaLM-2 [32] on multiple medical QA benchmarks, we experimented with various more basic queries without the noise of patient-specific document context. While Meditron-7B was able to minimally respond to queries like "List the most commonly used antidepressants to treat major depression" by listing the correct medications, it uniformly failed to compose any meaningful responses to our more complex clinical trend queries. This performance gap could be explained by the nature of the medical QA task targeted by medical benchmarks like MedQA and MedMCQA that Meditron-7B was trained on [5], in which the LLM was evaluated on fact-recall medical-school style exam questions that do not demand proficiency in integrating unstructured data to compose fluent responses as our ClinicalTrendQA queries require. Thus, we elected to use **GPT-3.5-Turbo** due to its demonstrably superior ability among the three evaluated LLMs to construct coherent responses citing and explaining relevant retrieved context.

THIS PAGE INTENTIONALLY LEFT BLANK

Chapter 7

Conclusion

Although there has been prior work on retrieval-augmented clinical question-answering, a significant barrier to adopting clinical RAG in practice is its lack of security mechanisms to protect patient medical data. To address the challenge of securing retrieval with authorization control, we propose the **federated secure RAG (FS-RAG) architecture** for flexibly integrating user authentication and policy-based access control mechanisms into a hierarchically federated retrieval system. We implemented OIDC authentication servers and retrieval-step access control policies for a federated 3-hospital case study. Across all 20 evaluated clinical trend queries and 8 experimental user access profiles in the case study, we verify that the FS-RAG implementation, in comparison to the federated insecure baseline, correctly restricts the most relevant retrieved document set based on user access privileges. Thus, we show that our proposed FS-RAG model maintains retrieval correctness under the constraint of guaranteeing unauthorized documents are never leaked.

7.1 Future Work

Our implementation is a proof-of-concept to demonstrate the core principles of federated access control. As such, future work is necessary to productionalize the system for hospital deployment. In particular, our implementation uses the simpler but less secure symmetric HS256 algorithm to sign tokens during OIDC authentication. Future iterations should switch to the asymmetric **RS256** signing algorithm to avoid the dependency of storing and sharing secret keys between the central RAG app and all hospital authentication servers. Similarly, while we did not consider network security for simplicity, it is important to migrate from our implementation's insecure HTTP communication to **HTTPS** on future deployment.

We primarily focused on enforcing access authorization during retrieval. The related problem of **protecting sensitive data in access-authorized documents** was not investigated in the scope of this project. We believe **differential privacy** is an additional interesting area to explore for its potential privacy guarantees. We can employ differential privacy mechanisms to inject finely-calibrated noise into the retrieved document content after authorization. As a result, patient-identifying statistics irrelevant to the query — such as date of birth, age, race, and other demographic information — are never exposed outside of the local leaf retriever knowledge base. Perturbing sensitive but irrelevant retrieved document data would ensure that such private data is not leaked to the generation LLM, central RAG app, and other `RouterRetriever` nodes (during document aggregation) in the federation while minimizing impact on response quality. If noising retrieved document information using differential privacy schemes leads to unacceptably poor generation accuracy, **multi-party computation** techniques like those proposed in **Private-RAG** [39] represent another promising avenue to explore collective sharing of knowledge bases without noise-perturbing the document content by modeling the problem of retrieving the top-k relevant documents as a k-nearest neighbors search across an inverted file index (IVF).

Scalability of our system as the number of participating hospitals n increases is also an important area for future work. Our case study involved only three hospital servers, so future work should conduct a performance analysis of our system for larger networks of hospital servers (i.e. $n = 10, 20$) to better understand our scalability bottlenecks. Considering the FS-RAG architecture and implementation, we have identified potential bottlenecks in the system for future scalability analysis:

1. **Qdrant Vector Store Creation Time.** We observed that the initialization

step of constructing Qdrant vector stores from raw EHR data contributed substantial latency to the first query to our system. Fortunately, the vector stores are persisted and thus initialization only needs to be performed once unless the document embedding model changes. However, Qdrant overhead in adding, deleting, or modifying EHR document data after initialization is a crucial concern due to the extremely dynamic nature of patient medical records that our project does not explore in-depth.

2. **Federated Retrieval Latency.** Compared to a centralized RAG system, FS-RAG introduces **retrieval-step overhead** due to performing PBAC checks for entry to each retriever and access to each retrieved document at leaf retrievers. Federation mechanisms like extra computation in aggregating documents are also potential sources of retrieval latency. Network load is an additional consideration as n scales, since federating retrieval relies on HTTP(S) requests from the central RAG app to each hospital server.

3. **Response Context Window.** Our choice of $k = 10$ documents was sufficient for the limited quantity of data in our case study, but practical applications of FS-RAG across larger hospital networks may demand larger k to ensure sufficient relevant context is given to the generation LLM. Scalability in k is primarily limited by the context length of the LLM, which constrains the size in tokens of the combined prompt (k-documents plus query). GPT-3.5-Turbo [4] has a context length of 16K tokens, which can accommodate $k \approx 80$ documents of chunked size 800 characters, so we do not anticipate it being the primary bottleneck of our system.

Finally, we note that while this project did not have the resources to incorporate human evaluation, future work with ClinicalTrendQA should consider employing licensed physician input when generating and assessing ground-truth responses to improve benchmark robustness and demonstrate alignment with established medical understanding.

THIS PAGE INTENTIONALLY LEFT BLANK

Bibliography

[1] Hiba Ahsan, Denis Jered McInerney, Jisoo Kim, Christopher Potter, Geoffrey Young, Silvio Amir, and Byron C. Wallace. Retrieving Evidence from EHRs with LLMs: Possibilities and Challenges. *arXiv preprint arXiv:2309.04550*, 2023. https://doi.org/10.48550/arXiv.2309.04550.

[2] Emily Alsentzer, John Murphy, William Boag, Wei-Hung Weng, Di Jindi, Tristan Naumann, and Matthew McDermott. Publicly Available Clinical BERT Embeddings. In Anna Rumshisky, Kirk Roberts, Steven Bethard, and Tristan Naumann, editors, *Proceedings of the 2nd Clinical Natural Language Processing Workshop*, pages 72–78, Minneapolis, Minnesota, USA, June 2019. Association for Computational Linguistics.

[3] Claudio A. Ardagna, Sabrina De Capitani di Vimercati, Sara Foresti, Tyrone W. Grandison, Sushil Jajodia, and Pierangela Samarati. Access control for smarter healthcare using policy spaces. *Computers & Security*, 29(8):848–858, 2010.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language Models are Few-Shot Learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[5] Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, Alexandre Sallinen, Alireza Sakhaeirad, Vinitra Swamy, Igor Krawczuk, Deniz Bayazit, Axel Marmet, Syrielle Montariol, Mary-Anne Hartley, Martin Jaggi, and Antoine Bosselut. MEDITRON-70B: Scaling Medical Pretraining for Large Language Models. *arXiv preprint arXiv:2311.16079*, 2023. https://doi.org/10.48550/arXiv.2311.16079.

[6] Chung, David Ferraiolo, David Kuhn, Adam Schnitzer, Kenneth Sandlin, Robert Miller, and Karen Scarfone. Guide to Attribute Based Access Control (ABAC)

Definition and Considerations. Special Publication (NIST SP), National Institute of Standards and Technology, Gaithersburg, MD, 2019-08-02 2019. https://doi.org/10.6028/NIST.SP.800-162.

[7] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022. https://doi.org/10.48550/arXiv.2210.11416.

[8] de Carvalho Junior, M. A., & Bandiera-Paiva, P. Health Information System Role-Based Access Control Current Security Trends and Challenges. *Journal of healthcare engineering*, 2018(6510249), 2018. https://doi.org/10.1155/2018/6510249.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[10] Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. RAGAs: Automated evaluation of retrieval augmented generation. In Nikolaos Aletras and Orphee De Clercq, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta, March 2024. Association for Computational Linguistics.

[11] Ana Ferreira, David Chadwick, Pedro Farinha, Ricardo Correia, Gansen Zao, Rui Chilro, and Luis Antunes. How to Securely Break into RBAC: The BTG-RBAC Model. In *2009 Annual Computer Security Applications Conference*, pages 23–31, 2009.

[12] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997*, 2024. https://doi.org/10.48550/arXiv.2312.10997.

[13] Goldberger, A. L., Amaral, L. A., Glass, L., Hausdorff, J. M., Ivanov, P. C., Mark, R. G., Mietus, J. E., Moody, G. B., Peng, C. K., & Stanley, H. E.

PhysioBank, PhysioToolkit, and PhysioNet: components of a new research resource for complex physiologic signals. *Circulation*, 101(23):E215–E220, 2000. https://doi.org/10.1161/01.cir.101.23.e215.

[14] J. Marc Overhage, MD, PhD, and David McCallie Jr., MD. Physician Time Spent Using the Electronic Health Record During Outpatient Encounters. *Annals of Internal Medicine*, 172(3):169–174, 2020. PMID: 31931523.

[15] A. Johnson, L. Bulgarelli, T. Pollard, S. Horng, L. A. Celi, and R. Mark. MIMIC-IV (version 2.2). PhysioNet, 2023. https://doi.org/10.13026/6mm1-ek67.

[16] A. Johnson, T. Pollard, S. Horng, L. A. Celi, and R. Mark. MIMIC-IV-Note: Deidentified free-text clinical notes (version 2.2). PhysioNet, 2023. https://doi.org/10.13026/1n74-ne17.

[17] Johnson, A.E.W., Bulgarelli, L., Shen, L. et al. MIMIC-IV, a freely accessible electronic health record dataset. *Sci Data*, 10(1), 2023. https://doi.org/10.1038/s41597-022-01899-x.

[18] Johnson, Jeff and Douze, Matthijs and Jégou, Hervé. Billion-Scale Similarity Search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2021.

[19] Minki Kang, Jin Myung Kwak, Jinheon Baek, and Sung Ju Hwang. Knowledge Graph-Augmented Language Models for Knowledge-Grounded Dialogue Generation. *arXiv preprint arXiv.2305.18846*, 2023. https://doi.org/10.48550/arXiv.2305.18846.

[20] Kuhn, D. Richard and Coyne, Edward J. and Weil, Timothy R. Adding Attributes to Role-Based Access Control. *Computer*, 43(6):79–81, 2010.

[21] LangChain. www.langchain.com/.

[22] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[23] Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[24] Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan McDonald. On faithfulness and factuality in abstractive summarization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1906–1919, Online, July 2020. Association for Computational Linguistics.

[25] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[26] Alexey Romanov and Chaitanya Shivade. Lessons from natural language inference in the clinical domain. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1586–1596, Brussels, Belgium, October-November 2018. Association for Computational Linguistics.

[27] Sandhu, R.S. and Coyne, E.J. and Feinstein, H.L. and Youman, C.E. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[28] Santos-Pereira, C., Augusto, A.B., Correia, M.E., Ferreira, A., Cruz-Correia, R. A Mobile Based Authorization Mechanism for Patient Managed Role Based Access Control. *Information Technology in Bio- and Medical Informatics*, 2012. `https://doi.org/10.1007/978-3-642-32395-9_5`.

[29] Kwangsoo Seol, Young-Gab Kim, Euijong Lee, Young-Duk Seo, and Doo-Kwon Baik. Privacy-Preserving Attribute-Based Access Control Model for XML-Based Electronic Health Record System. *IEEE Access*, 6:9114–9128, 2018.

[30] Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval Augmentation Reduces Hallucination in Conversation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[31] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. Large Language Models Encode Clinical Knowledge. *Nature*, 620:172–180, 2023.

[32] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, Mike Schaekermann, Amy Wang, Mohamed Amin, Sami Lachgar, Philip Mansfield, Sushant Prakash, Bradley Green, Ewa Dominowska, Blaise Aguera y Arcas, Nenad Tomasev, Yun Liu, Renee Wong, Christopher Semturs, S. Sara Mahdavi, Joelle Barral,

Dale Webster, Greg S. Corrado, Yossi Matias, Shekoofeh Azizi, Alan Karthikesalingam, and Vivek Natarajan. Towards Expert-Level Medical Question Answering with Large Language Models. *arXiv preprint arXiv:2305.09617*, 2023. https://doi.org/10.48550/arXiv.2305.09617.

[33] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open Foundation and Fine-Tuned Chat Models. *arXiv preprint arXiv.2307.09288*, 2023. https://doi.org/10.48550/arXiv.2307.09288.

[34] Xintao Wang, Qianwen Yang, Yongting Qiu, Jiaqing Liang, Qianyu He, Zhouhong Gu, Yanghua Xiao, and Wei Wang. KnowledGPT: Enhancing Large Language Models with Retrieval and Storage Access on Knowledge Bases. *arXiv preprint arXiv.2308.11761*, 2023. https://doi.org/10.48550/arXiv.2308.11761.

[35] Yubo Wang, Xueguang Ma, and Wenhu Chen. Augmenting Black-box LLMs with Medical Textbooks for Clinical Question Answering. *arXiv preprint arXiv:2309.02233*, 2023. https://doi.org/10.48550/arXiv.2309.02233.

[36] Wang, C., Ong, J., Wang, C. et al. Potential for GPT Technology to Optimize Future Clinical Decision-Making Using Retrieval-Augmented Generation. *Ann Biomed Eng*, 2023. https://doi.org/10.1007/s10439-023-03327-6.

[37] Cyril Zakka, Akash Chaurasia, Rohan Shad, Alex R. Dalal, Jennifer L. Kim, Michael Moor, Kevin Alexander, Euan Ashley, Jack Boyd, Kathleen Boyd, Karen Hirsch, Curt Langlotz, Joanna Nelson, and William Hiesinger. Almanac: Retrieval-Augmented Language Models for Clinical Medicine. *arXiv preprint arXiv:2303.01229*, 2023. https://doi.org/10.48550/arXiv.2303.01229.

[38] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, Hao Zhang, Joseph E Gonzalez, and Ion Stoica. Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt,

and S. Levine, editors, *Advances in Neural Information Processing Systems*, volume 36, pages 46595–46623. Curran Associates, Inc., 2023.

[39] Guy Zyskind, Tobin South, and Alex Pentland. Don't forget private retrieval: distributed private similarity search for large language models. *arXiv preprint arXiv:2311.12955*, 2023. https://doi.org/10.48550/arXiv.2311.12955.