Clinical Question-Answering over Distributed EHR Data

by

Emily Jiang

S.B. in Computer Science and Engineering at 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: Emily Jiang
Department of Electrical Engineering and Computer Science
May 10, 2024

Certified by: Lalana Kagal
Principal Research Scientist
Thesis Supervisor

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

Clinical Question-Answering over Distributed EHR Data

by

Emily Jiang

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

Electronic health records (EHRs) have become standard in US clinical practice. However, the distributed, dynamic, private, and jargon-dense nature of medical data is a barrier in harnessing Large Language Models (LLMs) for the domain. Retrieval-augmented generation (RAG), in which an LLM is provided with both the question and context returned by an external retriever, is a promising technique for addressing the unique qualities of clinical text. LLMs using RAG can answer questions about patient records without training on privacy-sensitive data; updated records can also be queried immediately without finetuning. By exposing the source documents that inform the model response, RAG enables greater physician interpretability as well as reduced hallucination, both of which are crucial for safe deployment in healthcare. This thesis presents FEDRAG, a retrieval-augmented clinical question-answering (QA) system for clinicians to explore trends in patient data across distributed storage. We introduce a novel hierarchical design for federated document retrieval, in which leaf nodes perform local similarity search while non-leaf nodes route queries based on access policies and aggregate documents returned by their children. We also create a dataset on clinical trends over the MIMIC-IV database for the evaluation of QA systems on EHR data. FEDRAG is implemented in Python as a federation of Flask servers using LangChain, the Qdrant vector database for retrieval, and GPT-3.5 Turbo for generation. We present a case study of three medical organizations, and find that the federation scheme results in no loss of quality against a centralized baseline. We explore the impact of resource accessibility among users with varying access permissions, observing that retrieval and generation quality degrade reasonably as document access is restricted. Finally, we evaluate performance in the key abilities required of RAG systems. We conclude that despite remaining challenges in achieving high retrieval quality and noise robustness, FEDRAG is effective at synthesizing clinical trends through information integration across EHR documents.

Thesis Supervisor: Lalana Kagal
Title: Principal Research Scientist

Acknowledgments

I would like to thank the Decentralized Information Group at CSAIL for introducing me to interesting and challenging research problems; the staff of 6.390 for the opportunity to serve as a teaching assistant during my MEng; Alice Chen for her extensive and invaluable collaboration on this project; and my thesis supervisor Lalana Kagal for her support and guidance in my research journey.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Electronic health records (EHRs), digital versions of patients' health data, have seen widespread adoption over the past decade. As of 2015, 96 percent of all non-federal acute care hospitals had adopted a certified EHR system to manage patient data [31]. EHRs unlock a wealth of opportunities for data-driven approaches to healthcare. For a single patient, consolidating their medical history is useful for searching and examining trends in their health. Across all patients in a practice, clinicians may be able to use the collective data to help diagnose and treat patients.

EHRs contain a variety of data. While there are structured elements such as demographic information and bloodwork results, the bulk of clinical data is unstructured—unannotated text written by a clinician, such as patient notes and discharge summaries [21]. Unstructured data is well-suited for large language models (LLMs), which generate textual responses given natural language prompts. However, clinical data is unique along a few important axes, making it challenging to directly feed into LLMs. Unlike common NLP training data such as classic literature or Wikipedia articles, patient medical data is highly dynamic, with new records created at a high frequency. There is also no centralized repository of health records as there are multiple competing EHR platforms, and different organizations would not share patient data with each other unless that patient transferred between practices [29]. Constructing a centralized repository would also present security risks. This means that EHR data is distributed in nature and must remain that way when used in NLP applications. In the United States, the use and disclosure of patient medical data is also strictly regulated under the HIPAA Privacy Rule [30]. Even if LLM providers sign the required Business Associate Agreement for HIPAA-compliant clients, there remain risks in yielding the management of privacy-sensitive data to a third party [13]. Finally, patient notes are syntactically idiosyncratic, dense with clinical jargon and acronyms. In short, **EHR data tends to be distributed, dynamic, private, and idiosyncratic**, necessitating specialized solutions for the clinical NLP domain.

An automated healthcare system must align with physician priorities to facilitate adoption. One particularly important aspect to physicians is *interpretability*, or how easily a model's decision-making process can be understood by humans, rather than being a "black box." Interpretability is especially crucial in the healthcare setting. Patients have a reasonable right to explanation of the logic involved when machine learning is used to inform their medical treatment, a right that is explicitly codified by the European Union Global Data Protection Regulation [11]. LLMs, which synthesize text based on representations of language learned from training on trillions of tokens, pose particular problems for interpretability. It is unclear what the "source" informing an LLM response is—or whether one exists. Additionally, LLMs have been known to *hallucinate* incorrect information when they have not been trained on the query domain [46], which can be dangerous in clinical applications. Indeed, a 2024 study on the ability of LLMs to substantiate their responses to medical questions found that even the most advanced model (GPT-4 with retrieval-augmented generation) failed to consistently corroborate its claims, with 30% of its individual statements unsupported and nearly half of its overall responses not fully supported [44]. We thus emphasize the need for LLMs used in the medical domain to produce responses that are not only true but also easily proven true.

1.1 System Objectives

This thesis details the design and implementation of FedRAG, a clinical question-answering (QA) system that allows medical professionals to ask questions about the health records of their patients. FedRAG's design is motivated by the goal of **automated inference over large quantities of health data**, coupled with the **high degree of explainability** required for machine learning models in this domain. There is demonstrated potential for the utility of such a service: a 2023 survey on the perception and utilization of AI tools in health care found that more than 1 in 10 health care professionals already use AI technologies such as OpenAI's ChatGPT as part of their practice, and half of the remaining 90% intend to adopt them in the future [38].

The overarching goals of this system are to **provide data federation and privacy while improving interpretability** of model responses for physicians. We are motivated by the following use case. A clinician practices at a hospital that uses an EHR system. When making clinical decisions, the doctor will often be informed by the patient's medical history, which may be too long to quickly summarize, particularly for a recurring patient. Existing methods already exist for automatic summarization of EHRs [42]. However, physicians may additionally want to identify broader trends in their patient data. Other than practicing physicians, this may also be of importance to physician-scientists and biomedical researchers, such as at research hospitals or other medical research groups. While professionals in this domain would already be aware of large-scale trends such as a new COVID-19 wave, our goal is to provide users with the ability to answer questions such as the following:

- *For patients admitted with major depression who were prescribed Celexa, has the medication had any impact on their behavior?*
- *A patient is exhibiting the following symptoms: [...]. For other patients at this practice who have reported similar symptoms, what has been the most common diagnosis?*

1.2 Overview of Contributions

The primary contributions of this thesis are threefold:

1. To the best of our knowledge, we present the first effort to use a retrieval-augmented LLM to synthesize trends over clinical data in distributed storage.

2. We implement a novel hierarchical design for relevant document retrieval in which leaf nodes are responsible for local document search and non-leaf nodes are responsible for routing decisions and aggregation of intermediate results.

3. We introduce a new clinical trend dataset for evaluation of QA systems across patient data for use in this and future work.

Chapter 2

Background

With the recent rise of LLMs, clinical NLP research has explored ways to extract the rich information contained within unstructured EHR data. In 2022, Agrawal et. al. demonstrated that LLMs such as GPT-3 are effective at zero-shot and few-shot information extraction from clinical notes [1]. Their work focused on disambiguating overloaded medical acronyms based on context, highlighting the ability of LLMs to perform tasks in an idiosyncratic field without being trained or fine-tuned for that domain, an ability that we will also harness in our work. In this section, we summarize notable work on adapting LLMs for clinical tasks.

2.1 LLMs for Clinical Knowledge

There have been many recent efforts to adapt LLMs for clinical tasks. Broadly, they fall under two categories: creation of a new domain-specific model by fine-tuning an existing one, and adapting a general-domain model to clinical tasks. Below, we outline related work in both categories.

With the growth of state-of-the-art LLMs to hundreds of billions of parameters, methods such as few-shot prompting have emerged as computationally inexpensive ways to fine-tune these models compared to end-to-end fine-tuning. A notable example of a clinical model fine-tuned using such techniques is Med-PaLM [39], a clinical instruction prompt-tuned variant of Flan-PaLM [10], which is itself an instructiontuned variant of PaLM [9]. Med-PaLM exhibits state-of-the-art performance on several medical question-answering benchmarks, and performs similarly to clinicians in long-form response generation. Despite excelling at clinical QA tasks, Med-PaLM's use case is distinct from ours as the model focuses on general medical knowledge (such as medical licensing exam questions) rather than trend extraction from real patient data. Additionally, while our system does use an existing LLM, Med-PaLM (and the further improved Med-PaLM 2 [40]) is an unsuitable choice as it is a limited-access model.

Also under the umbrella of fine-tuning for the medical domain is Meditron [8], an open-source extension of Llama-2 [41] by pretrained on a comprehensive medical corpus containing PubMed articles, abstracts, and medical guidelines. Meditron-70B achieves state-of-the-art performance on major medical benchmarks and is competitive with closed-source LLMs, outperforming GPT-3.5 and Med-PaLM and coming within 5% of GPT-4 and 10% of Med-PaLM 2. In light of this performance on clinical tasks, we consider and evaluate the smaller model, Meditron-7B, as an LLM for our system's generation module in Section 3.3.2.

Ahsan et. al. take the latter approach, using Flan-T5 XXL in a zero-shot setting to infer based on clinical notes whether a patient has or is at risk of a particular condition [2]. They address the concerns of hallucination and interpretability by explicitly prompting the LLM to provide a "step by step" explanation for its response based on the patient note. This resulted in high-quality, interpretable outputs as perceived by radiologists. However, hallucination is still an issue, as there is no guarantee that the explanation will contain accurate citations from the patient's note. This work is similar to ours in that it directly deals with EHR data and focuses on physician interpretability, but their use case is to diagnose a single patient rather than our goal of synthesizing trends across EHR data of many patients. Further, the EHR data is manually included in each prompt instead of originating from an initial document retrieval phase (see Section 2.2 below). However, given the work's usage of Flan-T5 for a related task, we also experiment with Flan-T5 for generation in our system.

2.2 Retrieval-Augmented Generation

Retrieval-augmented generation (RAG) [26] is a framework for improving the quality of LLM-generated responses by using external documents at query time to supplement the model's internal information. The architecture combines a pre-trained retriever (canonically Dense Passage Retriever [20]) with a pre-trained generator (canonically BART [25]). For each query, the retriever first identifies a fixed number of relevant documents before passing them to the generator, which produces an output conditioned on those documents as context. The relevant documents are selected from an external knowledge source, which often contains different (e.g. more up-to-date) text than the training data. Document relevance is typically determined using a nearest-neighbor algorithm such as Maximum Inner Product Search or cosine similarity in a vector store [26].

RAG shows potential in adapting LLMs for clinical applications. Traditionally, the clinical NLP community has developed fine-tuned versions of general models, such as BioBERT [24] (initialized from general BERT weights), ClinicalBERT [3] (initialized from BioBERT weights), and the aforementioned Med-PaLM and Meditron. These specialized models exist because medical text contains abbreviations and jargon that a general model like BERT or PaLM would not have seen during training. However, **RAG eliminates the need for explicit domain adaptation**, since the external knowledge source can be from a different domain from the training data. Indeed, allowing LLMs to retrieve information from medical textbooks [43] or even utilize a web browser [45] has resulted in improved performance on medical QA tasks. Our work instead would use EHR data as the external knowledge source to enable clinical QA over patient health records. RAG also allows for **knowledge update without fine-tuning**, since the external knowledge source can be updated over time to reflect more recent information. Health data changes rapidly as doctors see patients continuously, making inexpensive knowledge update crucial.

RAG's grounding of the generator output on the retrieved documents **reduces the likelihood of hallucinations**, as well as making them easier to detect. The generator step returns the highest-probability output for the query conditioned on each of the retrieved documents [26]. In practice, this means that we can examine the retrieved documents to determine which ones were the most influential in producing the response. Hallucination can be detected when the final output contradicts the relevant documents, which is much easier than verifying correctness of each response over all relevant data without RAG.

We thus use a RAG architecture for our clinical QA system. As a paradigm rather than a specification, RAG can be implemented with any combination of retrievers and generators, which allows for the design of a custom retriever for our federated clinical data use case, while making direct use of a state-of-the-art, general-purpose LLM (here, GPT-3.5 Turbo) for the seq2seq generator.

Chapter 3

Design

The objective of this thesis is to design and implement FedRAG, a federated RAG-based clinical question answering (QA) system that will allow medical professionals to ask natural language questions about the medical records of their patients. The system is intended to service clinicians treating many patients at potentially multiple practices, as well as researchers, who want to explore trends across patient data. Our design allows us to achieve a **shared LLM across data** of separate hospitals **without explicit sharing** or consolidation of that data. Thus, queries can potentially extract more statistically significant trends across a higher volume of patient data without restricting the data origin to a single practice.

3.1 System Overview

Figure 3-1 illustrates the system architecture. The flow of data through the system for each query is summarized as follows.

1. The clinician submits a prompt to the web interface (described in Section 4.5).
2. The web app client sends both the prompt and the clinician's credentials to the retrieval module, represented by the tree of nodes in Figure 1.
3. The retriever's documents are federated over the set of leaf nodes at the right-most side. Depending on the specific prompt, the prompt and credentials are propagated to the nodes that are likely to contain relevant documents.

4. The retrieval module returns the top k relevant documents to the client, which then sends the documents and original prompt as a single input to the generator (an LLM).
5. From the input, the generator returns both its synthesized response and source documents used to the client, which then displays these to the clinician.

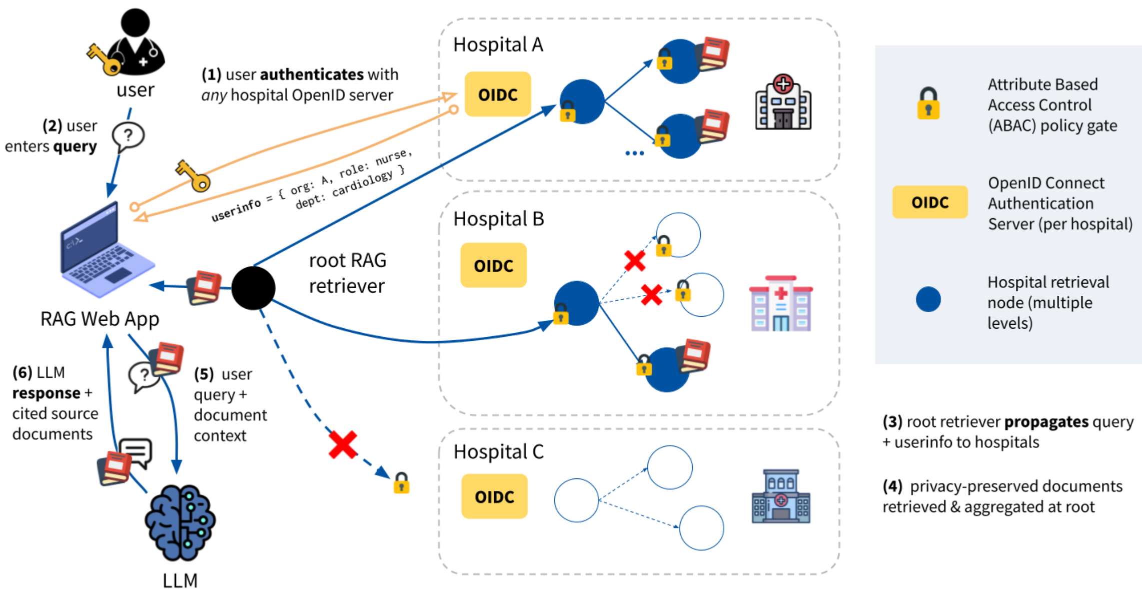

Figure 3-1: System architecture diagram.

Below, we describe the design of the retriever and generator.

3.2 Retrieval

As shown in Figure 3.1, the RAG retriever in our RAG system has a hierarchical structure. Each leaf node of the retrieval tree contains a subset of EHR data. Each non-leaf node routes queries to each of its children, where the query continues down the hierarchy if the user is permitted to access the data in that subtree. The partitioning of data among nodes is not random, but rather meant to reflect data provenance. The root node's children represent distinct medical centers, which will each have different children depending on the practice. For example, there might be separate child nodes for each of a hospital's ICU, Pediatrics, and Radiology departments. Our retriever design allows the federation scheme to be different across medical centers depending on how their data may already be stored; a hospital might for instance partition its data based on patient surname. The retriever structure is flexible: each node can have an arbitrary number of nested subdivisions.

3.2.1 Recursive Routing

The user query x begins at the root node and routes down the tree. Non-leaf nodes are responsible for routing the query to each of its children recursively until a leaf node is reached. However, a path down to a leaf may be terminated early at any point along the path if the user's credentials do not permit them to access the data contained in that node or its subtree. The user's credentials are obtained using the OpenID Connect authentication protocol at any organization in the federation, the details of which are not the focus of this thesis. The credentials should contain any identifying attributes needed to make access-control decisions, such as the user's organization, role, department, and email.

Access policies, implemented at the organizational level rather than at the system/root, are optional (an organization may choose to provide "open-source" data to all users) and attribute-based (users are allowed or denied based on the attributes in their credentials). Consider for example a query from a therapist about trends in anxiety disorder among her clients. The root node might have 10 children corresponding to the EHR data of 10 different organizations in the QA system. Based on her credentials, the therapist may only have access to the subset of hospitals or clinics that she practices in. At the next level of nodes, the query might continue onward only to those organizations' Psychiatry departments.

3.2.2 Retrieval at Leaf Nodes

Leaf nodes perform standard RAG document retrieval, where the knowledge base consists of the EHR data present at the leaf. Depending on each organization's data management, the scope of data at a leaf could be as coarse as an entire hospital's records or as granular as the records from patients diagnosed with clinical depression in one hospital's psychiatric department. To determine the top k documents, each leaf node conducts a similarity search between x and its documents. We use normalized ClinicalBERT embeddings to build the search space; as ClinicalBERT was fine-tuned on all unstructured data in MIMIC-III, these domain-specific embeddings should make the representation of the query and documents more meaningful in a clinical context.

The leaf returns the k most relevant documents and corresponding similarity scores to its parent.

3.2.3 Aggregation to Root

Each non-leaf node will aggregate all the documents from the children that it routed the query to. The documents are merged into a global sorted order of descending similarity score and truncated to the top k most similar across all its children, as shown in Figure 3-2. This comparison is valid even between different leaves as cosine similarity in normalized ClinicalBERT embedding space is used as the similarity score metric across all vector databases.

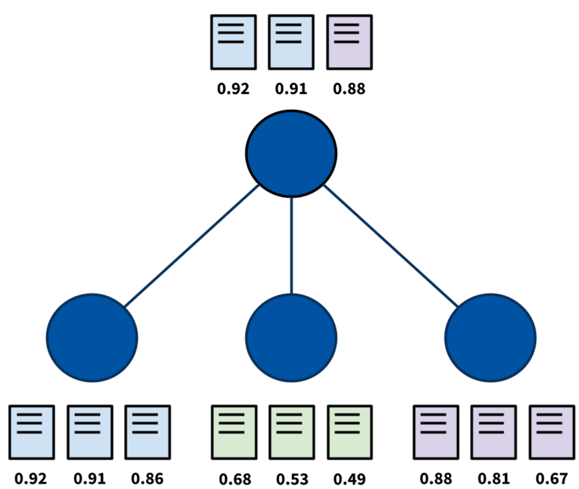

Figure 3-2: Visualization of aggregation of child documents at a parent node.

This process propagates back up the hierarchy, with each successive parent aggregating and truncating to the k best documents in its subtree. Finally, the root node returns the k most relevant documents overall to the QA service.

3.2.4 Improving Name Matching

We observed in early runs of the system that for questions naming specific patients, the vector store was often unsuccessful at returning documents from the correct patient, though it would return documents from other patients that were related to the query in a medical context. This implies that the ClinicalBERT embeddings—and likely other embeddings if used—did not consider the same name appearing in both the query and a document as meaningful to semantic similarity. As we felt that ClinicalBERT embeddings were still appropriate for our clinical use case, we added a retrieval optimization for improving name matching. We first use a named entity recognition system to detect names of patients in the query, then only consider documents from those named patients for retrieval, as further described in Section 4.2.1.

3.3 Generation

In this section, we describe the generation module of our RAG system, including the prompt and choice of generation model.

3.3.1 Prompt Template

We use the following prompt template for generation, in which *{context}* and *{question}* would be replaced with the list of retrieved documents in descending order of similarity score and the user's original query, respectively.

> *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 engineering is not a focus of this work. However, the above prompt was written with the intention of discouraging hallucinations; we also wanted to prevent the LLM from assuming that documents are relevant by mere virtue of being included even if they do not contain relevant content. For example, we observed that when a user with access to only admissions data asked about risk factors of stroke, the LLM would report that marital status and other fields in `admissions.csv` were risk factors, assuming without evidence that the irrelevant documents provided in the prompt must be from patients admitted for stroke. We discuss observed hallucinations further in Section 5.5.

3.3.2 Choosing an LLM

An LLM used for a trend-based question-answering system must be capable of producing a **fluent** and **coherent** response to the question, which

- **combines** information from all relevant documents in the context,
- **synthesizes** information about trends in its own words, and
- **cites** all documents used in its response.

In choosing an LLM for the generator module, there were several additional considerations. When working with clinical data, it may be beneficial to use a domain-finetuned model to obtain more clinically informed responses. However, one of the benefits of RAG is that it allows domain adaptation without finetuning—that is, we can make use of a general-purpose LLM to perform domain-specific tasks. We thus consider both general and clinical LLMs.

LLMs may also be either open-source or proprietary. We prefer open-source models as they can be run locally; proprietary models are typically hosted and accessed via API call, which may clash with HIPAA, GDPR, or institutional privacy restrictions depending on how patient health information is stored or used by the model host [16]. The two choices come with different costs: open-source models may require significant compute power to run locally, while proprietary models may incur costs for API calls.

Additionally, the utility of an LLM is bound by its *context length*, the maximum number of input tokens supported. Short context lengths heavily restrict RAG applications, as the retrieved documents may together be significantly longer than the original query. Longer context lengths allow more documents to be retrieved and leave more room for prompt engineering.

With these factors in mind, we selected three LLMs for consideration, and outline their comparison. Key information about these models is summarized in Table 3.1.

Table 3.1: Comparison of LLMs in key considerations.

Model	**Parameters**	**Context Length**	**Open**	**Domain-Specific**
Flan-T5 Large	783M	512	Yes	No
Meditron-7B	6.74B	2K	Yes	Yes
GPT-3.5 Turbo	Unknown	16K	No	No

Flan-T5 Large

Flan-T5 [10] is an instruction-finetuned version of the Text-to-Text Transfer Transformer Language Model (T5) [34] that achieves strong few-shot performance across many reasoning-based tasks. It is smaller than models with comparable performance and thus more efficient to run. Flan-T5 is publicly available on HuggingFace in five checkpoints of increasing size. We used the third largest model, Flan-T5 Large, as a compromise between size and performance, in order to easily run on an OpenStack instance for development.

Meditron-7B

As introduced in Section 2.1, Meditron is a suite of open-source LLMs adapted to the medical domain from Llama-2. The models are available in two sizes: Meditron-7B and Meditron-70B. Considering computational constraints, we focused on the smaller model, which outperforms its baseline Llama-2-7B on multiple medical reasoning tasks. We found that even Meditron-7B was too large to be be downloaded and run on our OpenStack instance. However, as we still wanted to compare responses from widely-used general LLMs with those from a state-of-the-art domain-finetuned LLM, we created a HuggingFace Dedicated Inference Endpoint [14] for the model that we could call from our instance. We ran our evaluation questions (Section 5.2) through the system with a custom dummy LLM that logged its input to a file rather than producing a response. These saved prompts were then fed into the HuggingFace inference endpoint to obtain Meditron-7B's responses.

GPT-3.5 Turbo

GPT-3.5 Turbo from OpenAI is a "set of models that improve on GPT-3.5 and can understand as well as generate natural language or code." We use the OpenAI API's default GPT-3.5 Turbo model, `gpt-3.5-turbo-0125`, which has a context window of 16,385 tokens and is more accurate at responding in user-requested formats compared to earlier models. Although OpenAI also offers the more powerful GPT-4 class of models, we opted for the more cost-effective model: `gpt-3.5-turbo-0125` costs $0.50 per 1 million tokens of input and $1.50 per 1 million tokens of output, while GPT-4 Turbo (considered to be more capable and less expensive than GPT-4) costs $10 and $30, respectively [5].

Using hosted LLMs such as GPT for clinical applications comes with important privacy considerations. In particular, GDPR and HIPAA compliance require zero data retention (ZDR) from services that handle personal health information [16]. While the OpenAI API already ensures that user input will not be used for training (the same guarantee does not apply to the online ChatGPT service), ZDR can be further enabled in eligible endpoints via support request. For these zero retention endpoints, OpenAI enables the signing of Business Associate Agreements (BAA), which is mandatory for customers who require HIPAA compliance [16]. While we do not use a BAA (as we do not represent a medical organization), OpenAI's support of HIPAA compliant applications is a promising justification for using GPT models in our system.

LLM Comparison

We observed very apparent differences in response quality between the three LLMs, and discuss our qualitative evaluation. To ensure a fair comparison, we used a maximum document length of 400 characters and set the number of documents retrieved (k) to 4 to accommodate Flan-T5 Large's context length, the smallest of the three. We present a representative example of each LLM's responses to the first question in our evaluation dataset (Table A.1):

> *For patients admitted to this practice who report heavy alcohol consumption, what are some of the most common diagnoses?*

The retrieval system returned the following four documents for the question:

1. For patient with name of Betty Henry: Elevated transaminases currently, but in classic 2:1 pattern for EtOH and given recent ingestion history, this is more likely the explanation. Medications on Admission: The Preadmission Medication list is accurate and complete.

2. For patient with name of Edward Fisher: Past Medical History: PAST PSYCH HX: No previous medication trials or psychiatric hospitalizations. One previous episode of being kept in a psychiatric ED in ___ in the context of having chest pain which turned out to be a panic attack. Gets occasional panic attacks (although not frequently enough to make a diagnosis of panic d/o). Has been seeing a counselor ___ at ___ x 4 months.

3. For patient with name of Cory Goodman: Service: MEDICINE. Allergies: No Known Allergies / Adverse Drug Reactions. Chief Complaint: multiple falls at group home. Major Surgical or Invasive Procedure: None . History of Present Illness: Mr. ___ is a ___ year

old male with a h/o psychosis admitted for falls over the past two days. Patient himself claims that people have been pushing him to the ground and beating him up.

4. For patient with name of Betty Henry: # **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** did not require detox on this admission. # HIV: CD4 282 on admission.

The responses of each LLM to the question and supporting documents are as listed in Table 3.2.

Table 3.2: Comparison of LLM responses to Q1

LLM	**Response**
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.)
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.

We observed that Flan-T5 Large produced either very short (often “yes” or “no”) responses or would provide citations of a single retrieved document. It seemed unable to paraphrase from the context or combine information from multiple documents. The model also would cite a document until reaching its maximum output length rather than stop outputting tokens at a sentence boundary. For instance, Flan-T5 Large's output in Table 3.2 quotes exactly the bolded text of document #4 above and does not end generation at a sentence boundary, nor does it include information from any of the other three documents. While Flan-T5 Large does produce relevant output, its inability to synthesize new ideas by combining separate sources of information renders it an ineffective choice for a trend-based question-answering system.

Meanwhile, Meditron-7B did not produce a coherent response at all, and was similarly unhelpful when answering other questions. In an attempt to reconcile the poor observed quality with the model's state-of-the-art performance on major medical benchmarks, we explored different prompts, including more basic questions without context. Results were mixed, with Meditron-7B correctly returning a list of antidepressants when asked what medications should be prescribed to patients with major depression, but still unable to answer other basic questions. For example, when asked "What are some symptoms of patients with seasonal affective disorder?" the model's entire output was the question repeated over and over. Given these results, we conclude that Meditron-7B is unsuitable for the generation module.

It is clear that GPT-3.5 Turbo produced the most coherent response of the three models, and also included in its response which parts of the context informed each component of its answer. Unlike Flan-T5, it was also able to integrate information from all four documents provided. It is worth noting that GPT-3.5 Turbo's response may still not be completely accurate in a clinical sense, as the question asks for *diagnoses* while the model response discusses some symptoms and observations instead such as elevated transaminases. However, we conclude that GPT-3.5 Turbo is proficient at the question-answering capabilities required for our application, and discuss its use in our generation module in Section 4.3.

Chapter 4

Implementation

We implemented FedRAG in Python. The structure consists of a "core" system, which implements the internal data types and utility functions necessary for federated retrieval, as well as wrappers and imports of LLMs for generation. The core system provides an API for both the central service and the organizations (e.g. hospitals) to build the hierarchical retriever system automatically based on their data.

Our implementation makes extensive use of LangChain, an open-source framework designed to simplify the creation of applications using LLMs [22]. A LangChain "chain" can support RAG applications by linking a text prompt with the output of a document retrieval module to feed as input to a generation module. The package additionally provides wrappers to work with common LLM application tools, notably the OpenAI API and Qdrant vector database.

We use Qdrant [33], an open-source vector similarity search engine and vector database, to store our document embeddings. We elected to use a vector *database* rather than a vector *library* such as FAISS [19] due to the highly dynamic nature of patient health records. Vector libraries produce immutable indexes and are optimized for static data; modifying the store would necessitate a full rebuild. In comparison, vector databases have full CRUD (create, read, update, and delete) support and are thus more appropriate for EHR data [7]. Among popular vector databases, we choose Qdrant because it is "tailored to extended filtering support," which is useful for our patient name filtering and other potential user filters [23].

4.1 Data

In this section, we describe the usage of EHR data in implementing three hospitals for our system. FedRAG's design does not require organizations to use any specific structure or organization of data, allowing this choice to be delegated to each organization, which may use different EHR management systems. Consequently, the data preprocessing methods for our case study are specific to the structure of MIMIC-IV data and not necessarily universal.

4.1.1 MIMIC-IV

We obtain our data from MIMIC-IV [18], a large contemporary EHR database. The de-identified database contains medications, lab measurements, surgeries, patient notes, and other health data from over 40,000 patients treated in the critical care units of the Beth Israel Deaconess Medical Center between 2008 and 2019. There is a rich body of clinical NLP literature using MIMIC-IV's predecessor MIMIC-III, but the MIMIC project webpage recommends new studies to use MIMIC-IV, as the older dataset will eventually be deprecated [28]. MIMIC-IV is freely accessible to researchers upon completion of a course on HIPAA requirements and signing of a data use agreement.

MIMIC-IV is separated into six modules to reflect the provenance of the data: Core, Hosp, ICU, ED (Emergency Department), CXR (chest X-rays), and Note. The last module contains deidentified free-text clinical notes, consisting of discharge summaries for hospitalizations and radiology imaging reports [17]. Thus, MIMIC-IV-Note is the source of unstructured EHR data, while the remaining five modules provide structured data. As both structured (e.g. blood oxygen level, length of hospitalization) and unstructured fields may help reveal trends, our system must support both types of data. In our case study setup, we use the admissions data (`hosp/admissions.csv`) and discharge summaries (`note/discharge.csv`), which are structured and unstructured respectively. While we could have included other types of structured data, we found that a large portion of structured data, such as patient vitals, lab results, and medications, are already listed in the corresponding discharge summaries and thus redundant. However, our system does support the inclusion of additional unstructured data outside of what was selected for our case study.

4.1.2 Preprocessing

MIMIC-IV provides a centralized store of EHR records from a single medical center. However, we need to process the data in order to mimic the federated structure that we expect among different medical centers.

Each MIMIC-IV module takes the form of a relational database. Patient data in tables across all modules are linked by the `subject_id` field representing a unique patient ID; all `subject_id` values are listed in `hosp/patients.csv` along with the corresponding patient's gender, (shifted) date of birth, and other demographic information. Since patients are only identified with a numerical `subject_id`, we use the PyPI `names` package to randomly generate a name for each patient based on listed gender; we did this both to more easily refer to individual patients and in an attempt to aid retrieval (see Section 3.2.4). We added the `name` column to all relevant CSVs with the patients' names wherever applicable.

Unstructured Data

We further preprocessed individual discharge summaries before adding them as documents in the RAG knowledge base. The discharge summaries in MIMIC-IV are semi-structured, in that they follow a general pattern of sections each containing unstructured notes written by a physician or a list, such as of medications or lab results. In particular, each discharge summary begins with the preamble in Figure 4-1.

Many header fields were scrubbed during MIMIC's de-identification procedure and would not be useful in determining document similarity. We thus removed the Name, Unit No, Admission Date, Discharge Date, Date of Birth, Sex, and Attending fields from the text of each discharge summary, and removed newlines between words of the same sentence (that had been added to maintain a line length limit in the database).

```
Name:  ___                    Unit No:   ___

Admission Date:  ___              Discharge Date:   ___

Date of Birth:  ___             Sex:   F

Service: MEDICINE

Allergies:
No Known Allergies / Adverse Drug Reactions

Attending: ___.

Chief Complaint:
fever

Major Surgical or Invasive Procedure:
none


History of Present Illness:
___ with history of morbid obesity, coronary artery disease,
```

Figure 4-1: Excerpt from the beginning of a MIMIC-IV discharge summary.

We then divided each note into sentences and created each document by appending as many consecutive sentences as possible without exceeding our maximum character length (800). The "service" field above was also parsed from each note and added as a new `service` column in `discharge.csv`.

In LangChain, document objects can be created using a document loader that reads an external data source such as a `.txt` file, `.csv` file, or webpage. LangChain provides a document loader `DataFrameLoader` for pandas DataFrames that requires specifying one of the dataframe's columns as the "text" column; the remaining fields will become the document metadata. In our use case, `discharge.csv` has the following columns: `note_id`, `name` (added in preprocessing), `subject_id`, `hadm_id`, and `text`, with the final column containing the unstructured text of the discharge summary. Although only the document text, not the metadata, is embedded and thus used to determine relevance to a query, the metadata can be helpful for debugging, evaluation, and interpretability (seeing exactly where the supporting evidence originated from). However, the metadata information may still be important in determining relevance between a document and query. This is especially true for the patient's name, which may be referenced by the user directly in queries over a specific patient's records; the remaining fields primarily serve as unique numerical identifiers. To include patient name in the text of each document, we prefix each chunk with:

For patient with name of {`name`}: {`chunk`}

An example of a retrieved document from unstructured data is shown in Figure 4-2, with the metadata dictionary above the document content; the metadata includes the cosine similarity score (`score`) of the document to some query.

{'hadm_id': 21975601, 'name': 'Carol Sellers', 'note_id': '10002013-DS-6', 'org': 'A', 'score': 0.8978011606555624, 'service': 'MEDICINE', 'subject_id': 10002013, 'text_index': 9}

For patient with name of Carol Sellers: Nuclear stress test showed a reversible, medium sized, moderate severity perfusion defect involving the left circumflex territory. In light of these stress test findings, the patient underwent cardiac catheterization. Prior to cardiac catheterization, she was prehydrated given her acute kidney injury. The patient had a drug-eluting stent placed to the obtuse marginal branch. The patient became acutely hypertensive during cardiac catheterization and was started on a nitroglycerin drip (see discussion below). The patient was weaned from this quickly. The patient was continued on aspirin (full-dose), plavix, metoprolol, and imdur. Her home dose of atorvastatin was increased given evidence of coronary artery disease.

Figure 4-2: Example of a document created from unstructured data.

The final documents and accompanying metadata are added to the vector database located at the appropriate leaf retriever.

Structured Data

The structured data in MIMIC-IV, unlike `discharge.csv`, does not have a primary column; instead, all the fields may contain equally important information. We thus wrote `AllColumnsDataFrameLoader`, a custom document loader class that includes the names and values of all fields in the document content. Assuming short field values in structured data, we create a single document from each row. Following the behavior of LangChain's `CSVLoader`, different field/value pairs are separated by newlines, and the metadata includes the row of the original DataFrame.

Figure 4-3 shows an example of a retrieved document from structured data; the newlines between field values are printed here as single spaces.

{'dept_id': 'admissions', 'org': 'A', 'row': 49, 'score': 0.7712185585507368, 'source': 'DataFrame'}

subject_id: 10002013 name: Carol Sellers hadm_id: 21763296 admittime: 2165-11-23 08:19:00 dischtime: 2165-11-26 15:40:00 deathtime: nan admission_type: DIRECT EMER. admit_provider_id: P072C5 admission_location: CLINIC REFERRAL discharge_location: HOME HEALTH CARE insurance: Other language: ENGLISH marital_status: SINGLE race: WHITE edregtime: 2165-11-22 20:54:00 edouttime: 2165-11-23 16:04:00 hospital_expire_flag: 0

Figure 4-3: Example of a document created from structured data.

4.1.3 Case Study Data

The case study we present for evaluation is of a federation of 3 organizations, namely Hospital A, Hospital B, and Hospital C. The data for Hospital A was obtained from the set of discharge summaries corresponding to the first 50 unique `subject_ids` in `hosp/patients.csv`. We used the `service` field added in preprocessing to split the summaries into 11 distinct services: cardiothoracic, medicine, neurology, neurosurgery, obstetrics/gynecology, orthopaedics, plastic, podiatry, psychiatry, surgery, and urology. Of these, we selected medicine (the largest service by far with 89 notes), surgery (15 notes), orthopaedics (8 notes), and psychiatry (5 notes) to be the 4 departments for Hospital A; this subset was chosen mostly based on quantity of notes and to reduce clutter from having having all 11 services represented in the leaf retrievers.

The data for Hospital B was created using the same procedure as Hospital A, but from the second 50 unique `subject_ids` in `hosp/patients.csv`. We selected the medicine (70 notes) and cardiothoracic (7 notes) services to serve as Hospital B's departments.

The data for Hospital C was created from the combined notes of the neurology and neurosurgery services that were assigned to Hospital A and Hospital B (but not used by those hospitals in the case study). These 17 notes were concatenated and renamed into a single neurology department under Hospital C.

Each of the 3 hospitals also has an `admissions` retriever, containing the data in `hosp/admissions.csv` (`subject_id`, `name`, `hadm_id`, `admittime`, `dischtime`, `race`, `insurance`, etc.) corresponding to all of the hospital admissions present in that hospital's leaf retrievers.

4.2 Retrieval Module

We describe here the implementation of core data types for the retrieval module. We implement four retriever classes: `RootRetriever`, `RouterRetriever`, `LeafRetriever`, and `BaselineRetriever`. Retrievers inherit from the LangChain `BaseRetriever` class and must implement the `_get_relevant_documents()` method to define custom logic for document retrieval.

4.2.1 Root Retriever

The root retriever, as the place where retrieval starts and ends, is the only retriever that is stored by the application rather than by member organizations. We implement it with the `RootRetriever` class, which takes the following fields:

- `hospital_retrieve_uris`: list of retrieval endpoint URIs for each of the organization servers
- `userinfo`: dictionary of user credentials
- `search_kwargs`: dictionary of search parameters

Our search parameters are set to `fetch_k`=20 and `k`=10 to specify that the 20 most relevant documents to the query should be initially retrieved from the vector store, but only the 10 most relevant documents should be returned from `_get_relevant_documents()`. Fetching more documents than are eventually returned is a heuristic to lessen the likelihood of returning fewer than k final documents due to many relevant docs failing the post-retrieval access control check at the leaf (Section 4.2.3).

Taking in the user query, the `_get_relevant_documents()` method first uses spaCy's named entity recognition (NER) to identify any people mentioned in the query. We observed some inconsistencies with whether the NER model correctly stripped the 's possessive from names (e.g. as in "Betty Henry's symptoms"), so we manually stripped out the 's instances. The list of `names` found, if nonempty, is stored as `search_kwargs["filters"]["name"] = names`. The filters will be used at the leaf level to build Qdrant retrieval filters.

The root then sends a HTTP GET request to each of its `hospital_retrieve_uris` with the arguments to the child retrievers' `_get_relevant_documents()` methods: `query`, `userinfo`, and `search_kwargs`. Each successful request returns a JSON containing a `docs` field with a (possibly empty) list of serialized LangChain `Documents`. If a request is unsuccessful (e.g. because a hospital server is down), the root retriever simply does not use data from that organization. The retrieved documents across all organizations are sorted by cosine similarity score (which is added to document metadata by Qdrant) and only the top k returned as described in Section 3.2.3.

4.2.2 Router Retriever

Organizations have a single parent node that serves as the root of the organization's subtree. Unless the organization chooses not to subdivide their data (in which case a single leaf retriever would be sufficient), that parent node would be a router retriever. A `RouterRetriever` is initialized with the following fields:

- `id`: string identifier that uniquely identifies this retriever within the organization
- `children`: list of pointers to the retriever's child nodes, which may be other `RouterRetriever`s or `LeafRetriever`s
- `abac_gate`: attribute-based access control policies for this retriever, expressed in the policy language of the `py-abac` package

The router retriever's `_get_relevant_documents()` method is straightforward. The `userinfo` is passed as an argument to the method; based on the credentials, if the access control policies allow the user to access the retriever, the retriever sorts and truncates the documents returned by calling `_get_relevant_documents()` on each of its children. If the user is not permitted to access the retriever, an empty list is returned instead of the top k documents.

4.2.3 Leaf Retriever

The leaf retriever represents the bulk of retrieval functionality as it is here that document embeddings are stored and queried. A `LeafRetriever` is initialized with the following fields:

- `id`: string identifier that uniquely identifies this retriever within the organization
- `db_path`: the path on the client side where the Qdrant database will be persisted
- `df`: pandas DataFrame containing the data stored at this leaf
- `text_col`: the column in `df` containing the text to be embedded; if not specified, `df` is assumed to contain structured data
- `abac_gate`: access control policies for access to this retriever ("right to search")
- `abac_pdp`: access control policies for access at the document level ("right to retrieve")

Given the above fields, the `LeafRetriever` creates a `vectorstore_retriever` property representing the Qdrant database. As a pydantic `cached_property`, the generation method is called once when the property is first used, and then cached. If the leaf contains unstructured data, the generation method first performs the chunking and name-prefixing on `df` described in Section 4.1.2. This part of the data preprocessing is done by the core system rather than by the organization because the organizations are only responsible for providing data, not how that data is embedded. Next, the appropriate document loader is called on the resulting DataFrame: `DataFrameLoader` for unstructured data and the custom `AllColumnsDataFrameLoader` for structured data. The `Document`s returned by the loader are used to initialize a Qdrant database located at `db_path` with ClinicalBERT as the embeddings.

The leaf retriever's `_get_relevant_documents()` method performs retrieval as shown in Figure 4-4. As in the router retriever, the user's credentials are checked against the `abac_gate` access control policies to determine if they have permission to search at this leaf. If this check fails, an empty list is returned immediately. Otherwise, we call `vectorstore_retriever.get_relevant_documents(query, filter, k)`, where:

- `query`: user's original question
- `filter`: a Qdrant filter, only created if `search_kwargs["filters"]` is nonempty
- `k`: number of documents to retrieve from Qdrant i.e. `search_kwargs["fetch_k"]`

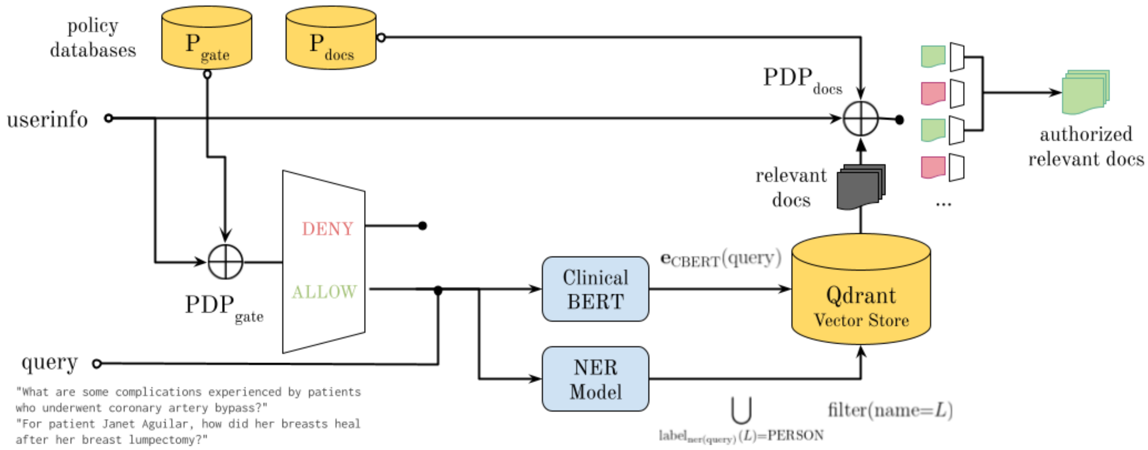

Figure 4-4: Diagram of the leaf retriever's `_get_relevant_documents()` logic.

Applications can modify the `search_kwargs["filters"]` dictionary at the root retriever to specify arbitrary filter conditions. (Specifically, our implementation filters on `name` metadata if a name was detected in the query.) The filters dictionary maps metadata attribute strings to lists of allowed values. The resulting Qdrant filter specifies that for every metadata attribute, the document's attribute value must be one of the allowed values. In other words, the filter is an AND over sets of OR conditions. For example, if two names appear in a query, the returned documents will only belong to patients with either of the two names.

After Qdrant returns `fetch_k` documents, the user's credentials are checked against the leaf retriever's `abac_pdp` policy decision point *for every document*; in this way, based on the document metadata and the organization's policies, a user might only have access to a subset of the `fetch_k` documents. Finally, the documents that pass this final access check will be sorted, truncated to `k` documents, and returned.

4.2.4 Baseline Retriever

The `BaselineRetriever` class is used only for evaluation purposes, in order to simulate a QA system over centralized (but possibly heterogeneous) data. It is similar to the `LeafRetriever` in that it creates a Qdrant database at a caller-specified location, and has an identical `_get_relevant_documents()` implementation. It is initialized with a list of `Documents`; for example, the result of appending the output of multiple document loaders, one for each data source included in the retriever.

4.3 Generation Module

Our generation module is a straightforward usage of LangChain's OpenAI API wrapper for the `gpt-3.5-turbo-0125` model. For evaluation purposes, we prefer the LLM responses to be deterministic given the same prompt, allowing for greater reproducibility as well as more accurate comparison between setups that should produce the same documents and responses (e.g. the centralized-insecure baseline and federated-insecure system in Section 5.3). OpenAI states that its model outputs are non-deterministic by default; however, we reduce non-determinism as much as possible by setting the model's `temperature` parameter to 0 upon initialization [6].

4.4 Organization Implementation

The goal was for the organizations to be responsible for as little logic as possible, since in a real system the organizations would be medical institutions such as hospitals that should be able to join the system easily and with minimal technical prerequisites. Thus, most organization code other than the boilerplate server setup code are calls to the core system's classes and methods. An organization in the federation must implement the remaining elements for itself, which we explain in greater detail below:

1. HTTP endpoint for retrieval
2. Authentication server for its users' credentials
3. Data storage and management
4. Retrieval subtree of router and leaf retrievers
5. Access control policies, if desired

Each organization should have a single parent node at the root of their subtree. In order for the organization to be reachable by the root retriever in the web application, its server must expose a `retrieve` endpoint that accepts HTTP GET requests. The arguments of the GET request (`query`, `userinfo`, `search_kwargs`) are the arguments to the parent node's `_get_relevant_documents()` method. The `retrieve` endpoint should return a JSON object with two fields: `query` and `docs`, which are the user's original query and the documents returned by retrieval in the organization's subtree, respectively.

An organization is additionally responsible for implementing its own OpenID Connect authentication server, deciding its own access policies, and deciding the structure of its retrieval subtree (i.e. how its data is partitioned among leaf and router retrievers). Since the system design is premised on federated data, an organization also needs to manage the storage of its own data, both raw data and Qdrant databases created by leaf retrievers.

Many different structures of organizations are supported by our system. For our case study in Section 5.1, we implement three example organizations in `/orgs` that share some skeleton code (`/orgs/auth_skeleton`) for the authentication server.

4.5 Web Application

The web application is written in Flask and contains code to authenticate the user using any trusted authentication server implemented by the hospitals in the federation. The authentication server will ask the user onscreen to consent to the sharing of their credentials (e.g. `{"org": "A", "role": "nurse", "dept": "icu"}`). Once the user is authenticated, the system will initialize a `RootRetriever` containing that `userinfo` dictionary and also specify search parameters for the entire system (e.g. `fetch_k`, `k`). For each query, those credentials will be forwarded to each retriever in the hierarchy in order to perform access-control checks against the resources.

The application code creates a LangChain chain that connects the retrieval and generation components of the system as follows:

1. The user question is passed as input to the root retriever.
2. The root retriever's returned documents (`context`) and the original question (`question`) are used to populate the prompt from the template in Section 3.3.1.

3. The prompt is passed as input to the LLM.
4. The LLM response is parsed and displayed to the user.

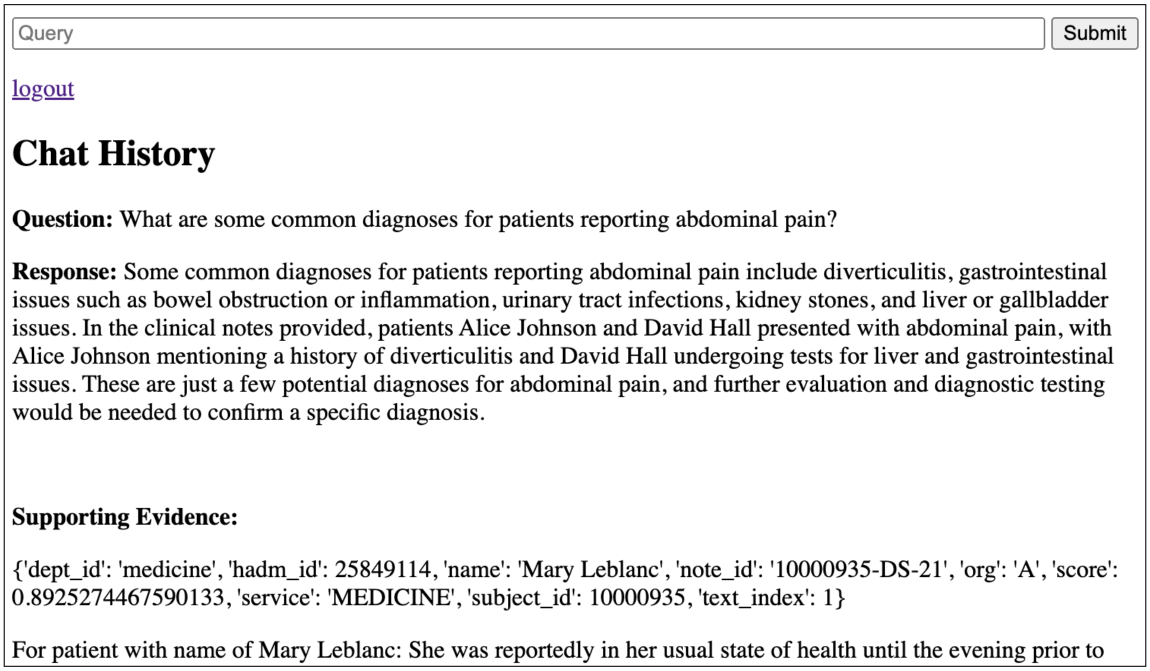

Figure 4-5: Interface for submitting questions and viewing responses and documents.

Figure 4-5 shows the user interface for submitting questions and viewing the system responses. When each question is submitted, the application will run the RAG chain on the question to produce a response, after which the question, response, and retrieved documents are displayed to the user. The user can then ask additional questions. All questions, responses, and documents will remain in the chat history until the user logs out.

Chapter 5

Evaluation and Discussion

The goal of our system is to identify trends over patient data that is distributed across numerous healthcare organizations. To this end, we evaluate FedRAG in multiple ways. We first demonstrate the usage of the system by constructing a case study with three organizations, in which each organization has users in different roles and departments, allowing varying degrees of access to resources in both a user's own organization and other organizations in the federation. Based on the subset of MIMIC-IV used in this case study, we construct the ClinicalTrendQA dataset of questions through manual observation of trends among discharge summaries and admissions data. We then evaluate the performance of the federated system compared to a centralized system containing the same data, as well as explore the impact of different access control restrictions on responses to the same question. In our comparisons, we consider both RAG response evaluation metrics and the supporting documents that were retrieved.

5.1 Case Study Setup

Our case study is a federation consisting of three organizations: Hospital A, Hospital B, and Hospital C. These hospitals have entered into an agreement to selectively share data within the federation and have designed their access control policies to reflect those data sharing rules. Each hospital implements its own Flask server. To simulate a distributed system, the FedRAG web application and the three hospitals each run on different ports of an OpenStack instance and communicate over HTTP as described in Chapter 4. The structure of router and leaf retrievers at each hospital is shown in Figure 5-1.

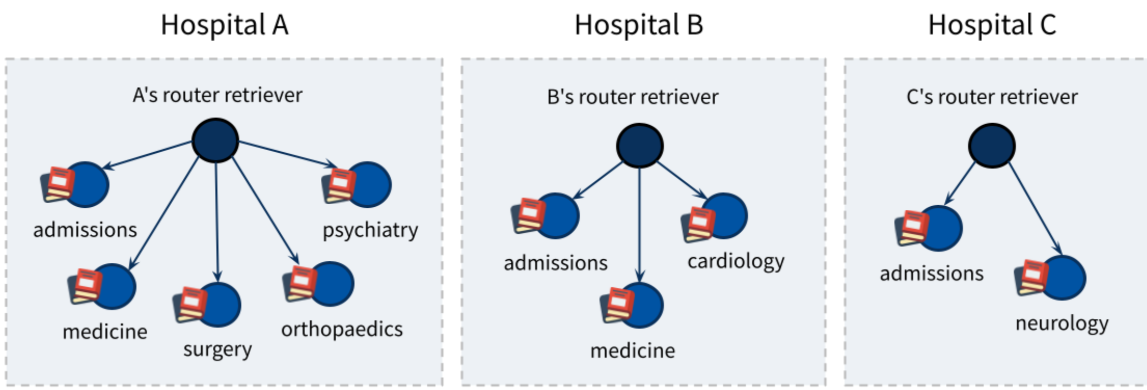

Figure 5-1: Organizational structure in the case study.

Hospital A is meant to represent a large general hospital with several more specialized departments such as psychiatry, while Hospital B represents a smaller medical center with fewer resources and specializing in cardiology. Hospital C, the smallest organization, is meant to represent a specialized research hospital focusing on neurology patients. Note that due to Hospital C's construction, a patient in Hospital A or B may also have records in Hospital C. This is fine given that patients of more general hospitals may seek neurological treatment in Hospital C; in fact, this allows for greater complexity in questioning, since asking about a single patient may require aggregation of documents across different organizations.

We also have the concept of *affiliations* to represent research groups and other miscellaneous attributes such as committee membership. A user's `userinfo` can thus optionally include a list of `affiliations`; specifically, having the `C_neuro` affiliation indicates membership in some neurology research group at Hospital C. This comes into play in our case study because a user part of Hospital A can also have the `C_neuro` affiliation, allowing them access to Hospital C's records that other users from Hospital A would be unable to retrieve.

5.1.1 Access Control

Each router retriever has to implement its own gate access control policies, while each leaf retriever has to implement both gate and document access control policies (see Section 4.2.3). In our case study, we did not make further access denials at the document level, which means that if a user was allowed to search a leaf retriever, they will also be allowed to retrieve any relevant documents found.

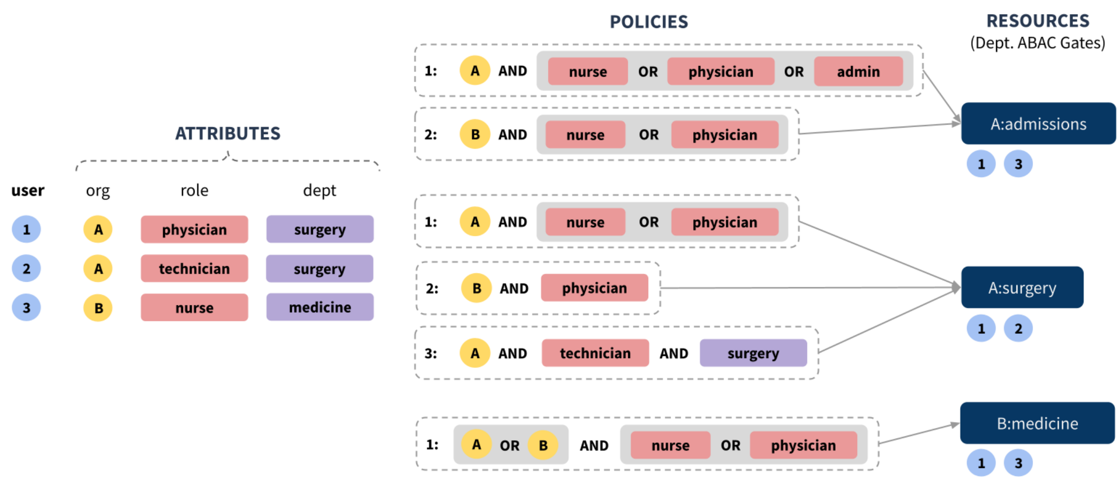

Figure 5-2: Visualization of select access policies between Hospitals A and B.

Our access control policies between Hospital A and B are designed with an imbalance such that Hospital A has the role of the "primary" hospital between the two. For example, physicians and nurses at Hospital A both have access to Hospital B's data, but only physicians (not nurses) at Hospital B have access to Hospital A's data. This policy design is for example purposes only—to allow more variations in document accessibility by role—and does not necessarily reflect a realistic setup between medical institutions. Figure 5-2 shows the access control policies for a subset of retrievers: Hospital A's admissions and surgery retrievers and Hospital B's medicine retriever. Within a single policy, represented by a dashed rectangle, each attribute may take on any of the specified values (OR), while conditions on different attributes must all be met (AND). If, like Hospital A's surgery retriever, access is controlled by multiple policies, being allowed by any of the policies is sufficient (OR).

5.1.2 Users

To evaluate the effect of different levels of document access on the application's response to the same question, we set up a variety of users under different organizations, roles, departments, and other attributes. We create 8 users in total; their `userinfo` attributes are listed in Table 5.1, where empty cells represent undefined attributes. Note that the **id** column is not part of our implementation and is only used to compactly identify the user scenarios in Table 5.3.

Table 5.1: Attributes of users in the case study.

id	name	org	role	dept	affiliations
U1	A.phys.neur	A	physician	surgery	C_neuro
U2	A.phys	A	physician	medicine	
U3	A.nurse	A	nurse	psychiatry	
U4	A.tech.rad	A	technician	radiology	
U5	A.admin	A	admin		
U6	B.phys	B	physician	cardiology	
U7	B.nurse	B	nurse	medicine	
U8	C.research	C	researcher	neurology	

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 5-3: Table showing whether users are allowed by each organization's retrievers.

The access policies defined at each hospital result in the case study users having access to hospital retrievers as in Figure 5-3, where green cells represent being allowed by the policy check at that retriever and red cells represent denial. Both router and leaf (abbreviated by its first 3 letters) retrievers for each organization are shown. Note that the `A.phys.neur` user is uniquely able to access all data in the system.

5.2 ClinicalTrendQA

We construct ClinicalTrendQA, a set of 20 questions about trends over MIMIC-IV data. We use the subset of data assigned to the three hospitals in the case study to manually identify trends, so that users in the case study can (with the necessary permissions) obtain the necessary documents to answer the questions. The questions target three scenarios: trends over a single note, trends over multiple notes from a single patient, and trends over multiple patients. Table 5.2 shows a question from ClinicalTrendQA for each of those categories, along with suggested relevant notes. Note that for the single-patient question the relevant notes are all for the same (Tracy Thompson's) `subject_id`, while the `subject_id` varies in the relevant notes for the multi-patient question.

Table 5.2: Sample ClinicalTrendQA questions.

Trend Type	**Example Question**	**Relevant Notes**
Single note	Has the prescribed medication impacted Edward Fisher's behavior, such as reported incidents of outbursts or other impulsive actions?	10000883-DS-4
Single patient	Describe the progression of left-sided weakness for Tracy Thompson.	10003299-DS-4, 10003299-DS-6, 10003299-DS-7
Multiple patients	For patients admitted to this practice who report heavy alcohol consumption, what are some of the most common diagnoses?	10000826-DS-17, 10002930-DS-10, 10005024-DS-4...

While relevant notes are provided for the sample questions above, we do not include "gold" documents as part of the actual dataset. This is due to a unique challenge of using RAG systems for trend-based question-answering. For example, a straightforward question like "What are some common medications used to treat major depressive disorder?" would have gold documents that directly state known facts. For single-note and single-patient questions, it may still be feasible to identify all relevant notes. However, for multi-patient questions—the focus of the work—there may not exist an objective list of most relevant notes, or the full list may be extremely large (i.e. for questions on very broad trends).

Likewise, we do not provide "ground truth" responses for the questions. Not only would writing gold responses require gold supporting documents, we felt that in the context of identifying trends in clinical notes, it would only be appropriate for gold responses to be written by physicians or clinical research scientists. As professional input was out of the scope of this thesis, we did not feel that it would be meaningful for the researchers, who have no clinical knowledge, to provide suggested responses. Rather, the questions are used as a means of comparison between responses produced by a centralized RAG system and `FedRAG`.

The full ClinicalTrendQA dataset is available in the Appendix.

5.3 Evaluating the Federated System

In evaluating the role of federation in our system, the main goal is to ensure that the RAG QA system over federated data suffers no loss of functionality or quality compared with an equivalent system over centralized data. We first establish a **centralized-insecure** baseline system (abbreviated **CI**), in which the retrieval module consists of a single `BaselineRetriever` as described in Section 4.2.4. The single retriever is initialized with all `Document` objects created from the data of all three organizations in the case study, and is preprocessed using the same methods (e.g. name prefixing, maximum chunk size). We also define the **federated-insecure** scenario (abbreviated **FI**) to be the FedRAG system in which no access controls are used. By construction, the vector database in the centralized baseline contains exactly the documents of all vector databases in the federated system combined; this detail is crucial to our evaluation.

In our evaluation, we also discuss the impact of varying degrees of document accessibility on the quality of retrieval and generation. We define each user's scenario to be the FedRAG system from their access perspective. For instance, the **A.phys.neur** scenario has access to all retrievers (and thus all documents) according to Figure 5-3 and should be informationally equivalent to the **federated-insecure** scenario, while the **A.admin** scenario would only have access to Hospital A's admissions records. We would thus expect both the relevance of retrieved documents and quality of generated responses to be greater for **A.phys.neur** than for **A.admin**. The baseline for comparison in all user scenarios is the **federated-insecure** scenario, as the latter is informationally equivalent to an all-powerful user who has permission to access every document in the system.

5.3.1 Retrieved Document Intersections

For each question in ClinicalTrendQA, we compute the intersection between the k documents returned by the **federated-insecure** system and those returned by the **centralized-insecure** baseline. We expect this intersection to be equal to 1 for all questions as there should be neither loss of information nor difference in similarity ranking between the centralized and federated system with equivalent data.

We also explore the "power" of each case study user by calculating the retrieved document intersection between that user scenario and the **federated-insecure** system. A user with higher power would have an intersection closer to 1, since it would have access to a greater percentage of the most relevant documents overall (i.e. the set returned by the **federated-insecure** scenario). We refer to this metric as `ixn` and report its value for each question under all scenarios in Table 5.3, as well as each scenario's average over all questions. To conserve space, the column names are shorthand for the scenario, with **FI** indicating **federated-insecure**, **U1** indicating **A.phys.neur**, and so on according to the id to name mapping in Table 5.1.

The intersections are as expected given each user's retriever access in Figure 5-3, indicating the functionality of the federated system. Specifically, the `ixn` for the **federated-insecure** and **A.phys.neur** (U1) scenarios is always 1 because both scenarios have access to the same information as the centralized baseline and will thus always retrieve the same most relevant documents.

Considering the relatively powerful (but not omnipotent) user **A.phys**, with access to all of Hospitals A and B, we see that the `ixn` is often 1, but is 0 for questions 7 and 12, which both refer to patients in Hospital C—indeed, due to the name filter A.phys will retrieve 0 docs for those questions. Meanwhile, this user has `ixn` $= 0.5$ for Question 13, in which the named patient Tracy Thompson is the only patient to have documents in multiple organizations (A and C). The average `ixn` of A.phys is 0.845, suggesting that the user has access to the majority of the resources needed to answer the evaluation questions. Note as well that the values for the A.phys and A.nurse scenarios are equivalent as the two users have the same retriever access.

Table 5.3: Retrieved document intersections between each scenario and its baseline.

Q#	**FI**	**U1**	**U2**	**U3**	**U4**	**U5**	**U6**	**U7**	**U8**
1	1	1	1	1	0	0	1	0.5	0
2	1	1	0.8	0.8	0	0	0.8	0.4	0.2
3	1	1	1	1	0	0	1	0.5	0
4	1	1	1	1	0	0	1	0.3	0
5	1	1	1	1	0	0	1	0	0
6	1	1	1	1	0	0	1	0	0
7	1	1	0	0	0	0	0	0	1
8	1	1	1	1	0	0	1	0	0
9	1	1	1	1	0	0	1	0	0
10	1	1	1	1	0.4	0	1	0	0
11	1	1	1	1	0	0	1	0	0
12	1	1	0	0	0	0	0	0	1
13	1	1	0.5	0.5	0	0	0.5	0	0.5
14	1	1	0.9	0.9	0	0	0.9	0.1	0.1
15	1	1	1	1	0	0	1	1	0
16	1	1	1	1	0	0	1	1	0
17	1	1	0.9	0.9	0.1	0	0.9	0.1	0.1
18	1	1	1	1	0	0	1	0.5	0
19	1	1	0.9	0.9	0	0	0.9	0.3	0.1
20	1	1	0.9	0.9	0	0	0.9	0.3	0.1
Avg	1	1	0.845	0.845	0.025	0	0.845	0.25	0.155

For the less permissioned users **A.tech.rad** and **A.admin**, we see that they both have very low average `ixn`. Note that the latter's score of 0 for every question does not mean that **A.admin** is *never* able to retrieve any of the k most relevant documents, but rather that none of Hospital A's admissions records were in the top k most relevant documents in the entire system for any of the 20 evaluation questions. However, given that question 17 ("Is Medicare or Medicaid more widely used for insurance among patients at Hospital A?") was written specifically to target Hospital A's admissions retriever, it appears that the ClinicalBERT embeddings are not fully aligned with our subjective notions of relevance.

5.3.2 ROUGE and BLEU Scores

ROUGE [27] and BLEU [32] are standard LLM evaluation metrics for QA systems. ROUGE is designed for summarization systems and focuses on recall. ROUGE describes a class of metrics including ROUGE-N, the overlap of n-grams between the system and reference response, and ROUGE-L, the longest common subsequence between the system and reference response. For our evaluation, we use the most common **ROUGE-1**, measuring the overlap of unigrams (individual words), as well as **ROUGE-L**. In contrast, BLEU is typically used to evaluate machine translation models and measures the precision of n-grams with respect to a reference, adding a brevity penalty to discourage short responses from scoring highly based on precision alone. While the summarization focus of ROUGE is a closer fit to our use case than machine translation for BLEU, both metrics are commonly used together in evaluating QA systems. Both ROUGE and BLEU scores range between 0 and 1, with higher scores indicating higher similarity between the system response and the reference.

For each evaluation scenario, we compute the ROUGE-1, ROUGE-L, and BLEU metrics and report the average score over all questions in Table 5.4, with the average intersection μ_{ixn} from Table 5.3 included again for reference. In each of the three metric columns, the maximum value is in **bold** while the minimum value is in *italics*.

We observe that the most powerful user **A.phys.neur** scores the highest in all metrics while the least powerful (according to `ixn`) **A.admin** scores the lowest in all metrics. This is in line with our expectation that the lack of information available to the latter user results in low quality generator responses when compared to the full-access **FI** baseline. To explore further the relationship between information access and response quality, we plot ROUGE-1 values against `ixn` for the five user scenarios with the highest variance in `ixn` across questions in Figure 5-4.

Table 5.4: 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.tech.rad	FI	0.03	0.40	0.28	0.16
A.admin	FI	0	*0.35*	*0.25*	*0.10*
A.nurse	FI	0.85	0.63	0.55	0.43
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

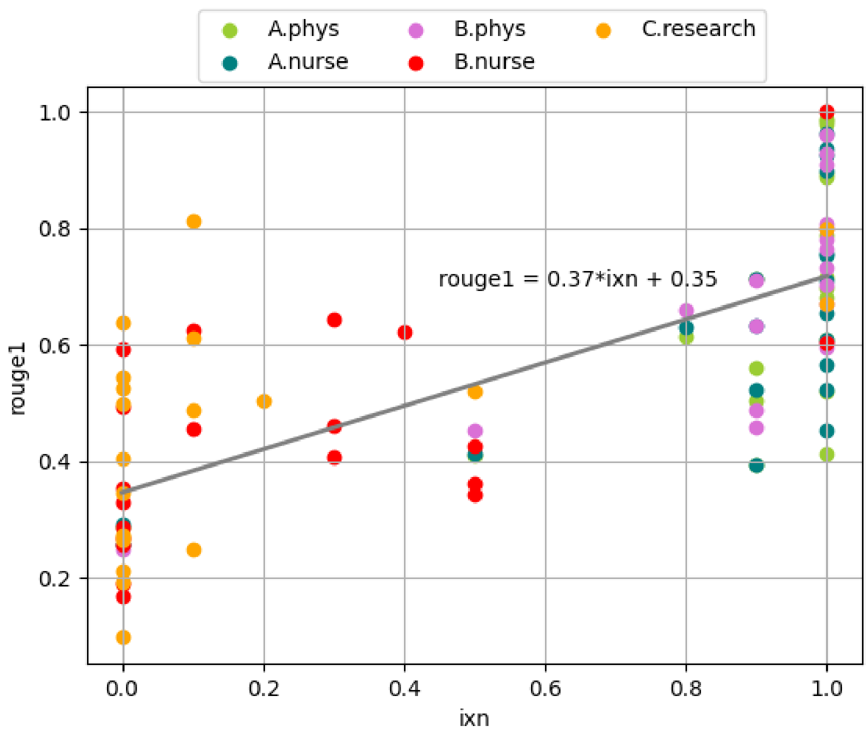

Figure 5-4: Plot of ROUGE-1 against `ixn` for 5 highest `ixn` variance scenarios.

The plot has a positive trendline but is noisy, especially for `ixn`s of 0 and 1 due to the high number of data points with those values. We only plot the scenarios with greater variance in an effort to reduce noise, as any ROUGE scores not equalling 1 for `ixn` = 1 should be entirely due to the nondeterminism of the LLM; that is, when identical documents are retrieved (and sorted in descending order of relevance), the prompt to the LLM is identical to that used in the baseline. For the `ixn` $= 0$ case, low ROUGE scores are likely noise from some word overlap with the baseline response; high ROUGE scores may signify hallucination of an answer close to the reference without use of the same context.

5.4 Quality Scores

We evaluate retrieval using the *context relevance* metric, a quality score used in contemporary evaluation practices for RAG systems that measures the precision and specificity of retrieved documents [15]. We use the Context Relevancy metric from the RAGAS [12] evaluation library, which uses the following formula [36]:

$$\text{context relevance} = \frac{|S|}{|\text{All sentences in retrieved context}|} \tag{5.1}$$

Above, $|S|$ is the number of sentences within the context that are absolutely required for answering the given question, determined by prompting GPT-3.5 Turbo.

We evaluate the generation quality of the overall RAG system with the standard quality scores of *answer faithfulness* and *answer relevance* [15]. The former assesses the factual consistency of the generated response against the retrieved context, while the latter measures the relevance of the response to the original question. These metrics are ideal for our system evaluation as they are both reference-free (i.e. do not require gold documents or responses). We again use the RAGAS library to compute the two generation metrics using LLM-based adjudication. To calculate the faithfulness score, RAGAS first prompts an LLM to break the response into a set of individual claims; the LLM is then prompted to determine whether each claim can be inferred from the retrieved context. The final score is given by the following formula [37]:

$$\text{faithfulness} = \frac{|\text{Claims in response that can be inferred from given context}|}{|\text{Claims in response}|} \tag{5.2}$$

The answer relevance metric is defined as the mean cosine similarity between the original question and a set of artificial questions generated based on the system response. Conceptually, this metric is based on the assumption that an answer that correctly addresses the question will allow the original question to be reconstructed from it with high probability. Rather than measuring factuality (which is covered by the faithfulness metric), answer relevance penalizes responses that are incomplete or contain redundant details. In the formula below, E_o is the embedding of the original question, N is the number of generated questions (default 3), and E_{g_i} is the embedding of the ith generated question [35]:

$$\text{answer relevance} = \frac{1}{N}\sum_{i=1}^{N} \frac{E_{g_i} \cdot E_o}{\|E_{g_i}\| \, \|E_o\|} \tag{5.3}$$

Table 5.5: Average RAGAS Context Relevance, Faithfulness, and Answer Relevance scores between all scenarios and their baselines.

Scenario	**Context Relevance**	**Faithfulness**	**Answer Relevance**
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

The context relevance, faithfulness, and answer relevance quality scores calculated by RAGAS are averaged across all questions for each evaluation scenario in Table 5.5. Each column's maximum value is again in **bold** and the minimum in *italics*.

In Table 5.5 we note the relatively high values and narrow range of faithfulness scores; since faithfulness measures an LLM's ability to make claims consistent with the context, we believe this points to the consistently good performance of GPT-3.5 Turbo. Comparatively, our answer relevance scores have higher range and variance.

In particular, there is a strong positive correlation between a user's access to relevant information (i.e. μ_{ixn}) and their average answer relevance score, as shown in Figure 5-5. This is expected, since a lack of pertinent information would reasonably result in an answer less relevant to the original question.

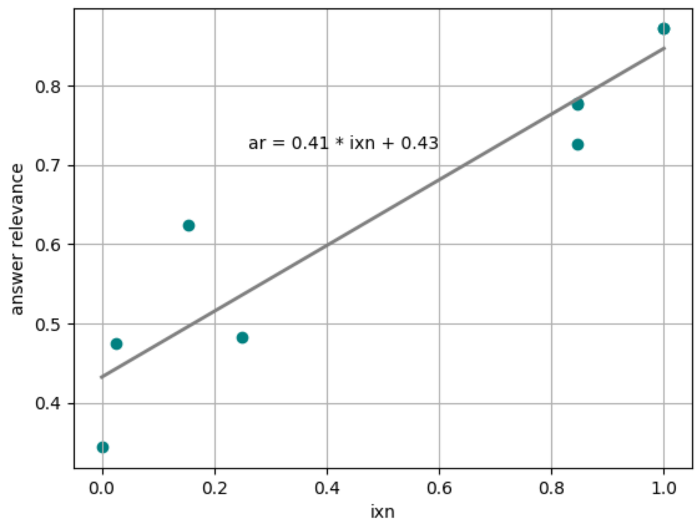

Figure 5-5: Plot of average answer relevance (`ar`) against `ixn` for all users. The equation of the linear regression line is $\texttt{ar} = 0.41 * \texttt{ixn} + 0.43$ with $R^2 = 0.89$.

To address the consistently low context relevance scores, we argue that the method used by RAGAS for calculating context relevance does not accurately reflect retrieval quality for our use case. Referring to Equation 5.1, RAGAS calculates context relevance on a sentence-level granularity: first a set of all sentences is obtained by parsing each retrieved document into its sentences, then the LLM is tasked to extract the sentences from the set that are absolutely required to answer the question. However, while RAGAS may be suitable for questions that can be answered with individual facts (their context relevance example asks "What is the capital of France?" [36]), our target trend-based questions require context from discharge summaries that often spans multiple sentences. When considered individually, a document's constituent sentences may not each be recognized as independently relevant by the LLM. Thus, we propose that in future work on RAG systems over EHR data, a more coarse-grained context relevance calculation, such as document-level, should be employed.

5.5 Required Abilities

We discuss qualitative observations of the system's ability to perform the following:

- *information integration*: the ability to synthesize knowledge from multiple documents into the response
- *noise robustness*: the ability to manage noise from documents that may be related but do not provide substantial information
- *negative rejection*: the ability to refrain from responding when the retrieved context does not contain the information needed to answer the question

These are three of the four major required abilities used in RAG system evaluation; the fourth, *counterfactual robustness*, does not apply here as we assume that all patient EHR data is factually correct [15].

We observe that the system is **effective at information integration**. This capability is introduced in Section 3.3.2, in which we demonstrate the ability of GPT-3.5 Turbo to *combine* and *cite* information from all 4 retrieved documents. In Table 5.6 below, we demonstrate FEDRAG's ability to integrate information not only across documents but also across organizations. The most relevant documents to Q13 are located in both Hospital A's medicine retriever (A_{med}) and Hospital C's neurology retriever (C_{neu}). In the table, we present users with all 4 combinations of access or denial to A_{med} and C_{neu} and their responses to Q13. We annotate the responses such that information derived from A_{med} is in bold red font; information from C_{neu} is in blue italics. We see that A.phys.neur includes content from both retrievers; A.phys and C.research each include information accessible to them from A_{med} and C_{neu} respectively; B.nurse, with access to neither retriever, correctly reports that it has insufficient information. While the source organization of each document is unknown to the generator, these findings still reflect that the system meets our goal of synthesizing trends across federated, cross-organization data.

We next evaluate noise robustness and negative rejection in tandem. These required abilities are related as they both refer to an LLM's ability to assess the relevance of the retrieved context. If the response indicates a false assumption that

Table 5.6: Annotated responses from select users on Q13 demonstrating information integration and negative rejection.

User	A_{med} Access	C_{neu} Access	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.

irrelevant documents are relevant for answering the question, this should be considered a type of hallucination. Negative rejection appears to be a special case of noise robustness in which *all* retrieved documents are noise; in this scenario, the LLM is expected to state that it has insufficient information to answer the question.

We conclude that FedRAG is **generally proficient at negative rejection**. In Table 5.7, we compare the responses from user A.phys, who has access to all relevant documents (`ixn`=1), and B.nurse, who does not have access to any relevant documents (in fact, no documents would be returned due to the name filter), to Question 8 ("What have been the prior causes of abdominal pain for patient Alice Johnson?"). We see that the former user answers correctly based on the context while the latter user correctly reports that "there is insufficient information" as the "notes do not mention" relevant information.

Table 5.7: Select responses of users with and without relevant doc access for Q8

User	**Access**	**Response**
A.phys	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. The abdominal pain she experienced was located in the lower left quadrant and was associated with diverticulitis flare-ups. Additionally, she presented with RLQ abdominal pain, which was similar to her previous episode of diverticulitis but on the other side of her abdomen.
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.

We observed that the LLM was able to identify cases of insufficient information in the majority of occurrences. However, in Table 5.8, we present a case of hallucination in which the user A.tech.rad retrieves 0 documents for Question 13, yet reports a fictional narrative of the progression of Tracy Thompson's left-sided weakness unsupported by any context. This occurs despite this addition to the prompt (Section 3.3.1): "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." We recognize that such a hallucination, which makes up facts about patient medical history, can be dangerous in a clinical setting.

Table 5.8: Select responses showing instances of LLM hallucination.

User	Q#	Excerpt from 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...
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...

Finally, we observe that the system **struggles with noise robustness**. Lack of noise robustness is often observed through a different class of hallucinations. In this class, the LLM, rather than using solely its parametric memory to generate an unsupported response as aforementioned, instead falsely incorporates information from irrelevant documents. This may occur when either some or all of the retrieved documents are irrelevant; in the latter case, this hallucination undermines the system's negative rejection ability as well. For example, in Table 5.8, A.admin can only retrieve patient admissions data such as Figure 4-3 to answer Question 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?"). Yet the response incorrectly cites the irrelevant admissions documents. The hallucination appears to stem from the LLM's "assumption" that the documents provided are relevant despite no such evidence in the document text. Specifically, Nancy Jones was indeed admitted multiple times to the emergency room and Carol Sellers was admitted for an "observation" admission type according to the context, but there is no evidence that the admissions were for orthopedic issues or occurred after falls.

Poor noise robustness can occur in spite of good information integration. For example, when selecting an LLM in Section 3.3.2, GPT-3.5 Turbo is able to combine information from all documents retrieved. However, closer inspection of the 4 context documents reveals that #2 and #3 both do not specifically mention heavy alcohol consumption (the subject of Question 1). A system with a high degree of noise robustness should be able to avoid using those documents to synthesize its response. Ultimately, the numerous instances of poor noise robustness are indicative of both limitations in the generator's reasoning capabilities as well as poor retrieval quality, and thus may be improved via either component of the RAG system.

Chapter 6

Conclusion

In this thesis, we present FedRAG, a system for using retrieval augmented generation to build a question answering service over federated and access controlled data. The work is motivated by a clinical use case, in which medical professionals ask natural language questions about trends over patient EHR data. We describe the novel design of the hierarchical retrieval module and its delegation of functionality between the root, leaf, and intermediate nodes. After detailing the process of selecting an LLM for the generation module, we describe the preprocessing of the MIMIC-IV dataset to be used in our evaluation case study. We outline the implementation of the retrieval module using LangChain and Qdrant; the generation module using the OpenAI API; and the organizations and web application using Flask.

In our evaluation, we first describe the setup of our case study with 3 organizations and 8 users with different levels of document access, and present our evaluation dataset ClinicalTrendQA. Using the dataset, we evaluate and discuss the efficacy of the federated system in comparison to a centralized system with the same data. We also compute the same set of metrics for each of the 8 users in comparison to the fully-permissioned federated system to assess the impact of document access on retrieval and generation quality. Finally, we discuss qualitative observations about the system's performance in the required capabilities of RAG systems.

Overall, while the retrieval results based on ClinicalBERT cosine similarity may not be fully value-aligned with physicians' notions of relevance, we show that retrieval is equally functional in the federated and centralized settings. The system still sometimes produces incorrect or hallucinated responses when given only irrelevant context; however, these responses can be fact-checked by the end user against the retrieved documents. Limitations in system deployability, methodology, and the scope of trends that can be identified are discussed in Section 6.1. We ultimately conclude that FedRAG is effective at identifying trends across federated EHR data with high interpretability, as desired.

6.1 Limitations

Below, we discuss limitations of the work along with possible solutions.

6.1.1 Supported Questions

While our work aims to answer trend-based queries, there remain limitations in the types of questions the system can reliably answer. Our retrieval scheme, which only performs a straightforward embedding similarity search, is unlikely to be effective at answering questions about longitudinal trends. As we only match for the most semantically similar documents, there is no guarantee that they will be distributed across time. Further, MIMIC-IV's de-identification process unlinks the discharge summaries from their date (see Figure 4-1). Date information is still recoverable, since each discharge summary is associated with a unique hospital admission, which is associated with an admit and discharge date in `/hosp/admissions.csv`. However, the retrieval system does not currently have the ability for complex reasoning *across* different leaf retrievers that would allow both the note and its corresponding `/hosp/admissions.csv` data to be recognized as relevant. Rather, it is unlikely that admissions data will be returned at all in the presence of any unstructured data, given its low semantic relevance to most natural language queries (see Figure 4-3). Indeed, for the only evaluation question written based solely on admissions data (#17 in Table A.1), not a single admissions document was retrieved except by user `A.admin` (who *only* had access to admissions data).

Thus, identifying time-based trends over MIMIC-IV data could only be made possible by augmenting the discharge summary chunks with temporal information during preprocessing. We can first use a SQL JOIN operation to add the time information from the admissions table as a column in the discharge summary table, and then manually add the time to each document's text, similar to how we prefix chunks with patient names in Section 4.1.2. Supporting longitudinal trends would likely further require specific retrieval optimization and filtering; for example, quotas based on document date/time ranges to ensure enough samples from different times throughout the history. We anticipate such case-specific handling to be quite complex and leave the implementation to future work.

FedRAG is also ineffective at answering SQL-like queries, especially ones that involve counting or other aggregate computations. For example, a question like "*How many* patients at this practice were admitted for pain following a mechanical fall last week?" would be extremely difficult for the system to answer correctly due to the counting component, in addition to the complexity of relative time. The main bottleneck for aggregation questions is the limit on total documents retrieved; while the limit can be increased, it is not feasible to have k on the same order of magnitude as the dataset size, which would potentially be required to answer large-scale aggregate questions. Future work on partial summarization at the router as suggested in Section 6.2 may enable aggregate computations.

6.1.2 Methodology

We recognize additional limitations in the implementation and evaluation. The organization data in our case study, and consequently the ClinicalTrendQA questions, focus on a relatively small and early subset of MIMIC-IV `subject_id`s. Thus, the full breadth of MIMIC-IV data may not be represented by our results. Despite this, we think that the subset of discharge summaries used is representative of the larger dataset, as `/note/discharge.csv` is sorted by `subject_id` rather than anything meaningful about their person or condition.

However, limitations remain in our evaluation. In clinical NLP research, automated evaluation metrics are often accompanied by human evaluation from licensed physicians, to compare the perceived utility of the model-generated and physician-generated responses. Our evaluation quality is thus limited as clinician feedback is out of scope for this thesis. Similarly, our evaluation questions were written without input from clinical professionals. The questions therefore may not be realistic for a variety of reasons including content, technical specificity, sentence structure, or grammatical correctness compared to what would be asked by intended users. Furthermore, the lack of gold documents prevents a quantitative evaluation of retrieval quality.

6.1.3 Scalability

Finally, scalability along multiple axes is not well explored. On the retriever side, the vector database is a bottleneck of scalability in the data size stored at leaf nodes. When initializing Qdrant databases using the same data split into different chunk sizes, we observed that smaller chunk sizes (i.e. more documents) resulted in significantly longer creation times, and likely search times as well. Future work should perform more thorough performance analysis. We additionally consider scalability in the number of organizations in the federation. Each of the organizations' retrieval endpoints are called in parallel, thus allowing for potential speedup over a centralized system by making parallel calls to smaller vector databases. However, the system could benefit from better fault tolerance when an organization server is slow or unresponsive. Currently, any HTTP errors including timeouts and failed requests are gracefully handled (by returning an empty list from that organization), but we do not implement our own timeouts for optimizing retrieval speed.

On the generator side, scalability in k may theoretically be limited by the context window of the LLM; however, we don't see this as a concern for OpenAI models. Our model, `gpt-3.5-turbo-0125`, has a context window of 16K tokens, while the premium `gpt-4-turbo` model accepts up to 128K [4]. If the generator were swapped out for one with a smaller context window, such as Flan-T5-Large (512 tokens), this could be a much more significant problem.

6.2 Future Work

Finally, we suggest avenues for system enhancement and future work. A primary axis for improvement is retrieval quality—making the returned documents more relevant. At the document level, we can explore using medical acronym expansion [1] on the discharge summaries to hopefully obtain more accurate embeddings. This is not guaranteed to be an improvement for retrieval, as the ClinicalBERT embeddings used for document similarity were trained on MIMIC-III notes, which likely contained similar acronyms. Still, acronym disambiguation may have additional benefits for the generator, allowing a non-domain-specific LLM to extract more meaning from the retrieved documents. In addition, tuning of the maximum chunk size parameter will likely affect retrieval quality. We hypothesize that shorter documents would be embedded with less "noise," thus resulting in more specific and relevant documents returned. This naturally trades off with performance; embeddings are the same length regardless of input text, so shorter chunks (i.e. more total documents) require strictly more time to initialize and query the vector database. We may additionally have to increase k to compensate for a shorter chunk size, so that the generator still receives a similar amount of context.

There is also potential in enhancing the router retrievers in the retrieval hierarchy. Router retrievers can be extended to perform summarization over the documents retrieved by its subtree. For example, n leaf retrievers each return k documents to its parent router retriever; that router, instead of doing the usual sort-and-truncate, can employ an LLM to summarize the information in the nk documents. It would then send the information learned from its subtree to its parent, which would summarize the information from its children, and so on. This has the benefit of allowing information from many more documents to make its way to the root retriever. Combined with additional SQL-like support, router summarization could also enable the aggregate computations discussed in Section 6.1.1 by performing the computation on each subtree and propagating the result upward. However, dealing with noise, error propagation, and loss of information through multiple rounds of summarization would likely be a challenge.

We can incorporate retrieval enhancements used in other RAG system studies as well. While similarity is traditionally measured between the document and query, recent work explores first generating (e.g. with an LLM) a hypothetical *response* to the original query, and then retrieving documents from the knowledge base based on similarity to the hypothetical response [15]. This method can potentially return more relevant documents in scenarios where the query and its correct response are semantically dissimilar. LLMs can also be used as helper tools post-retrieval by shortening or summarizing retrieved documents, or compressing the final prompt to optimize for LLMs rather than for human understanding [15].

Finally, the linked nature of EHR data suggests the use of knowledge graphs to enhance retrieval. For example, a single patient may link to multiple hospital admissions, each of which may link to multiple lab results. Consider documents containing structured records that do not have high embedding similarity with the query and thus would not be returned in the current scheme. Usage of knowledge graphs may allow those documents to still be retrieved as context, if they are linked by the graph to some other documents that *do* have high embedding similarity. In this way, incorporating a knowledge graph representation of EHRs may more thoroughly identify all relevant information needed to answer a question.

Appendix

Table 1: Full ClinicalTrendQA dataset.

#	**Question**
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?
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?
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?

Bibliography

[1] Monica Agrawal, Stefan Hegselmann, Hunter Lang, Yoon Kim, and David Sontag. Large Language Models are Few-Shot Clinical Information Extractors. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1998–2022. Association for Computational Linguistics, December 2022.

[2] 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.

[3] Emily Alsentzer, John R. Murphy, Willie Boag, Wei-Hung Weng, Di Jin, Tristan Naumann, and Matthew B. A. McDermott. Publicly Available Clinical BERT Embeddings. *arXiv preprint arXiv:1904.03323*, 2019.

[4] OpenAI API. Models. platform.openai.com/docs/models. Documentation.

[5] OpenAI API. Pricing. openai.com/api/pricing. Documentation.

[6] OpenAI API. Text generation. platform.openai.com/docs/guides/text-generation. Documentation.

[7] Erika Cardenas. Vector Library versus Vector Database. *Weaviate Blog*, December 2022. weaviate.io/blog/vector-library-vs-vector-database.

[8] Zeming Chen, Alejandro Hernández Cano, Angelika Romanou, Antoine Bonnet, Kyle Matoba, Francesco Salvi, Matteo Pagliardini, Simin Fan, Andreas Köpf, Amirkeivan Mohtashami, et al. Meditron-70b: Scaling medical pretraining for large language models. *arXiv preprint arXiv:2311.16079*, 2023.

[9] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. PaLM: Scaling Language Modeling with Pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[10] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling Instruction-Finetuned Language Models. *arXiv preprint arXiv:2210.11416*, 2022.

[11] William K Diprose, Nicholas Buist, Ning Hua, Quentin Thurier, George Shand, and Reece Robinson. Physician understanding, explainability, and trust in a hypothetical machine learning risk calculator. *Journal of the American Medical Informatics Association*, 27(4):592–600, July 2020.

[12] Shahul Es, Jithin James, Luis Espinosa-Anke, and Steven Schockaert. Ragas: Automated evaluation of retrieval augmented generation. *arXiv preprint arXiv:2309.15217*, 2023.

[13] Dan Fabbri. Using large language models like ChatGPT in healthcare: Make sure you understand the risks. *Imprivata*, June 2023. imprivata.com/company/press/using-large-language-models-chatgpt-healthcare-make-sure-you-understand-risks.

[14] Hugging Face. Inference Endpoints (Dedicated). huggingface.co/inference-endpoints/dedicated. Documentation.

[15] 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.

[16] Rafał Hofman and Izabela Pawlik. Large Language Models (LLMs) for Healthcare. Are They Secure?, February 2024. brightinventions.pl/blog/llms-healthcare-security.

[17] Alistair Johnson, Tom Pollard, Steven Horng, Leo A. Celi, and Roger Mark. MIMIC-IV-Note: Deidentified free-text clinical notes. *PhysioNet*, 2023.

[18] Alistair E.W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Tom J. Pollard Steven Horng, Sicheng Hao, Benjamin Moody, Brian Gow, Li wei H. Lehman, Leo A. Celi, and Roger G. Mark. MIMIC-IV, a freely accessible electronic health record dataset. *Scientific Data*, 10(1), 2023.

[19] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *arXiv preprint arXiv:1702.08734*, 2017.

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen tau Yih. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, Online, November 2020. Association for Computational Linguistics.

[21] Hyoun-Joong Kong. Managing Unstructured Big Data in Healthcare System. *Healthcare Informatics Research*, 25(1):1–2, January 2019.

[22] LangChain: langchain.com.

[23] LangChain. Qdrant. python.langchain.com/docs/integrations/vectorstores/qdrant. Documentation.

[24] Jinhyuk Lee, Wonjin Yoon, Donghyeon Kim Sungdong Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. BioBERT: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 09 2019.

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[26] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. In *Advances in Neural Information Processing Systems*, volume 33, pages 9459–9474. Curran Associates, Inc., 2020.

[27] 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.

[28] Medical Information Mart for Intensive Care. mimic.mit.edu.

[29] U.S. Department of Health and Human Services. Does a physician need a patient's written authorization to send a copy of the patient's medical record to a specialist or other health care provider who will treat the patient?, 2002. hhs.gov/hipaa/for-professionals/faq/271/does-a-physician-need-written-authorization-to-send-medical-records-to-a-specialist/index.html.

[30] U.S. Department of Health and Human Services. HIPAA for Professionals, May 2017. hhs.gov/hipaa/for-professionals/index.html.

[31] Office of the National Coordinator for Health Information Technology. National Trends in Hospital and Physician Adoption of Electronic Health Records. Health IT Quick-Stat #61.

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[33] Qdrant: qdrant.tech.

[34] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21:1–67, 2020.

[35] Ragas. Answer Relevance. docs.ragas.io/en/latest/concepts/metrics/answer_relevance.html. Documentation.

[36] Ragas. Context Relevancy. docs.ragas.io/en/latest/concepts/metrics/context_relevancy.html. Documentation.

[37] Ragas. Faithfulness. docs.ragas.io/en/latest/concepts/metrics/faithfulness.html. Documentation.

[38] Todd Shryock. AI Special Report: What patients and doctors really think about AI in health care. *Medical Economics Journal*, July 2023. https://www.medicaleconomics.com/view/ai-special-report-what-patients-and-doctors-really-think-about-ai-in-health-care.

[39] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620:172–180, 2023.

[40] Karan Singhal, Tao Tu, Juraj Gottweis, Rory Sayres, Ellery Wulczyn, Le Hou, Kevin Clark, Stephen Pfohl, Heather Cole-Lewis, Darlene Neal, et al. Towards expert-level medical question answering with large language models. *arXiv preprint arXiv:2305.09617v1*, 2023.

[41] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[42] Mengqian Wang, Manhua Wang, Fei Yu, Yue Yang, Jennifer Walker, and Javed Mostafa. A systematic review of automatic text summarization for biomedical literature and EHRs. *Journal of the American Medical Informatics Association*, 28(10):2287–2297, 2021.

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

[44] Kevin Wu, Eric Wu, Ally Cassasola, Angela Zhang, Kevin Wei, Teresa Nguyen, Sith Riantawan, Patricia Shi Riantawan, Daniel E. Ho, and James Zou. How well do LLMs cite relevant medical references? An evaluation framework and analyses. *arXiv preprint arXiv:2402.02008*, 2024.

[45] Cyril Zakka et al. Almanac: Retrieval-Augmented Language Models for Clinical Medicine. *Research Square*, May 2023.

[46] Muru Zhang, Ofir Press, William Merrill, Alisa Liu, and Noah A. Smith. How Language Model Hallucinations Can Snowball. *arXiv preprint arXiv:2305.13534*, 2023.