

To the Graduate Council:

I am submitting herewith a thesis written by Seyed Aran Arab entitled “Integrating Large Language Models and Graph Retrieval for Improved Decision-Making and Data Visualization in Electronic Health Records.” I have examined the final paper copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Industrial and Systems Engineering.

Dr.Xueping Li, Major Professor

We have read this thesis
and recommend its acceptance:

Dr.Bing Yao

Dr.Tom Berg

Accepted for the Council:

Amy Cathey

Vice Provost and Dean of the Graduate School

To the Graduate Council:

I am submitting herewith a thesis written by Seyed Aran Arab entitled “Integrating Large Language Models and Graph Retrieval for Improved Decision-Making and Data Visualization in Electronic Health Records.” I have examined the final electronic copy of this thesis for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Master of Science, with a major in Industrial and Systems Engineering.

Dr.Xueping Li, Major Professor

We have read this thesis
and recommend its acceptance:

Dr.Bing Yao

Dr.Tom Berg

Accepted for the Council:

Amy Cathey

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

Integrating Large Language Models and Graph Retrieval for Improved Decision-Making and Data Visualization in Electronic Health Records

A Thesis Presented for the

Master of Science

Degree

The University of Tennessee, Knoxville

Seyed Aran Arab

August 2025

© by Seyed Aran Arab, 2025
All Rights Reserved.

*In dedication to all of the people that supported me throughout both my career and
life.*

Acknowledgements

I would like to thank Emily because she supported me and pushed me through working on my thesis even when I was tired from work.

Abstract

Electronic Health Records (EHR) play a vital role in managing patient charts, encompassing both structured and unstructured data across various modalities. While EHR systems have already improved communication across medical facilities and addressed issues like poorly written notes, modern artificial intelligence techniques, especially Large Language Models (LLMs), offer even greater potential for enhancement. LLMs can process and integrate the multimodal nature of EHR data, enabling advanced applications in data management and analysis without sacrificing the foundational progress made by traditional systems.

In this paper, we propose to develop and implement advanced techniques such as Graph Retrieval Augmented Generation (GRAG) and Natural Language Processing (NLP) to transform data visualization within EHR systems. Our approach aims to provide healthcare providers, such as nurses and doctors, with a more intuitive and efficient means of accessing patient information, which is essential for rapid decision-making. Additionally, we are exploring the use of Large Language Models (LLMs) to optimize data entry tasks, thereby reducing the administrative burden on healthcare providers. By leveraging these AI-driven methods, we seek to foster critical thinking in clinical settings by automating repetitive tasks and enhancing the visualization of complex patient data. This research underscores the potential of LLMs and other AI techniques to further enhance the functionality of EHRs, ultimately contributing to a more effective and streamlined healthcare delivery system.

Table of Contents

1	Introduction	1
1.1	Technological Innovations in Healthcare	1
1.1.1	The History of EHR	2
1.1.2	Historical Struggles of EHR Systems	2
1.2	Towards an Intelligent EHR System	4
2	Literature Review	6
2.1	Electronic Health Records	6
2.2	Comparison Between SQL and GRAG Systems	6
2.2.1	Speed	7
2.2.2	Flexibility	7
2.2.3	Cost	8
2.2.4	Summary	9
2.3	LLMs	9
2.4	GRAG	10
2.5	Data Visualization	11
2.5.1	Graph Data Models in Healthcare	12
2.5.2	Graph Databases and Semantic Querying	12
2.5.3	Knowledge Graphs in Clinical Applications	13
2.6	Clinical Decision Support Systems (CDSS)	13
2.6.1	Limitations of Rule-Based Systems	14

2.6.2	LLMs for Dynamic Clinical Reasoning	14
3	Methodology	16
3.1	Graph Query and Data Visualization	18
3.1.1	Data Retrieval from Neo4j	18
3.1.2	Graphical Visualization of Admission Trends	19
3.1.3	Summary	19
3.2	RAG using Neo4j	19
3.2.1	Graph Data Extraction and Cleaning from Neo4j	19
3.2.2	Chronological Structuring and JSON Conversion	21
3.2.3	Embedding Patient Records into a Vector Space	22
3.2.4	Semantic Search and Similarity Scoring	23
3.2.5	Efficiency Considerations	23
3.2.6	Integration Summary	24
3.3	GRAG	24
3.3.1	Graph Construction	24
3.3.2	Graph Integration with LangChain	25
3.3.3	Language Model Configuration	25
3.3.4	Graph-Aware Question Answering	26
3.3.5	Response Generation	26
3.3.6	System Characteristics	27
4	Results	28
4.1	Graph Query	28
4.2	Example: Road to a Cure	30
4.2.1	Scenario	31
4.2.2	System Use	31
5	Conclusions and Future Work	37
5.1	Conclusion	37

5.2 Future Work	38
Bibliography	40
A Experimental Results	45
A.1 Images - Old Data - Correlation Matrix and Random Forest Results .	45
A.2 Images - Old Data - Neo4j Sample Queries in Gradio display	48
A.3 Images - Previous JSON Structures	50
A.4 Image - Neo4j - Patient Node and Diagnosis Node	53
Vita	55

List of Tables

3.1	Performance Metrics for multiple Embedding Models	24
-----	---	----

List of Figures

1.1	Sample image from the MedConnect Health Website MedconnectInc (2015) of their EHR system.	3
3.1	Hospital Workflow with GRAG and Graph Query Implementation . .	17
3.2	Neo4j database Node Types and Relationship Types	17
3.3	Graph-based data extraction and cleaning workflow from Neo4j to LangChain.	20
3.4	Hierarchical JSON structure after patient-wise chronological alignment.	21
3.5	Workflow of transformation from text document to vector representation using LangChain and FAISS.	22
3.6	Visual representation of query-to-vector match with similarity score annotation.	23
4.1	Google 'Busiest Times' graph for Walmart Alphabet (2024).	28
4.2	Hospital visitation trends by day of the week, compared to a specified date.	29
4.3	Graph-based estimated length of stay for patient Bobby Jackson. . .	30
4.4	Expanded Neo4j node for Jennifer Anthony, displaying interconnected nodes for diagnoses, lab tests, medications, and other clinical events. .	32
4.5	RAG output for the query: "When did Patient ID: 630439 take DENOSUMAB (Prolia) 60 mg/mL SYRINGE?"	33

4.6	GRAG output for the query: "What drugs are related to patient ID 630439?"	34
4.7	GRAG output for the query: "What drugs are related to patient ID 1573975?"	35
4.8	Generated Cypher query for inserting a temporal artery temperature reading into the Neo4j database.	36
A.1	Random Forest Algorithm R-squared and mean squared error	45
A.2	Random Forest Algorithm R-squared, Mean Squared Error, Mean Absolute Error, Root Mean Squared Log Error, and Root Mean Squared Error	46
A.3	Correlation Matrix for Random Forest Algorithm Generated for Healthcare Data	47
A.4	Sample of the model extracting information on all patients named Emily Johnson	49
A.5	Average stay time for Patient, Bobby JacksOn, taken from an average of the 20 most similar patients determined by the model	49
A.6	Initial JSON structure	51
A.7	JSON structure 2nd attempt	52
A.8	Patient node, George Woodward, in neo4j GRAG database	54
A.9	Diagnosis node, Asthma, in neo4j GRAG database	54

Chapter 1

Introduction

1.1 Technological Innovations in Healthcare

Technological innovation in healthcare has accelerated with the development of powerful tools such as cloud computing, digital twins, and artificial intelligence. In other industries, these technologies have revolutionized productivity, optimization, and decision-making [Wang \(2019\)](#). In healthcare, however, their integration remains fragmented and underutilized.

Cloud computing has seen adoption primarily as a cost-saving infrastructure layer, enabling hospitals and health systems to access software-as-a-service (SaaS) solutions without the overhead of local installations [Sultan \(2014\)](#). Still, [Griebel et al. \(2015\)](#) conducted a comprehensive literature review using the MEDLINE database and found limited penetration of cloud computing in areas such as Health Information Exchange (HIE), Electronic Health Record (EHR) systems, and clinical imaging. These gaps highlight the opportunity for transformative technologies to address healthcare-specific challenges—particularly in clinical decision support, data accessibility, and system interoperability.

As EHR adoption becomes widespread, the focus shifts from merely digitizing records to unlocking actionable insights from vast troves of patient data. This

paper explores the novel integration of Large Language Models (LLMs) and Graph Retrieval-Augmented Generation (GRAG) to enhance clinical decision-making and medical data visualization, bridging the divide between artificial intelligence capabilities and healthcare-specific needs.

1.1.1 The History of EHR

The widespread adoption of EHR systems in the United States began with the 2009 American Recovery and Reinvestment Act (ARRA), which allocated over \$38 billion toward the promotion of Health Information Technology (HIT), including EHRs [Cunningham \(2009\)](#). This legislative milestone catalyzed nationwide digitization efforts, laying the groundwork for innovation in how clinical data is stored, accessed, and analyzed.

However, early EHR systems primarily functioned as digital filing cabinets, with limited interactivity, poor usability, and minimal analytical capability. As healthcare providers became more reliant on digital documentation, the demand for smarter, more intuitive systems grew—especially those capable of interpreting unstructured clinical notes and supporting real-time decision-making.

With the rise of Natural Language Processing (NLP) and generative AI, there is a new frontier for EHR innovation. By leveraging the capabilities of LLMs and GRAG systems, it becomes possible to develop clinical interfaces that understand queries in natural language, retrieve relevant patient data, and even generate clinical insights—enhancing both physician efficiency and patient outcomes.

1.1.2 Historical Struggles of EHR Systems

Despite EHRs’ potential, widespread adoption has not been without its challenges. According to a comparative study by [McGinn et al.](#), major barriers to EHR implementation include technical limitations, poor usability, interoperability issues,

privacy concerns, and negative impacts on patient-provider interactions [McGinn et al. \(2011\)](#). An example of a current EHR system is shown in Figure 1.1

Technical and design limitations remain the most cited barriers, especially in resource-limited settings where computational costs and infrastructure constraints inhibit adoption [Feinstein \(1993\)](#). Recent advances in lightweight LLM architectures seek to address this, offering reduced model complexity while maintaining task performance [Nick Minaie \(2024\)](#). This enables implementation on lower-cost hardware and improves accessibility for underserved healthcare environments.

Perceived ease of use is another barrier. A survey by [Tubaishat](#) showed that while many nurses found EHRs helpful, a significant portion viewed them as cumbersome or

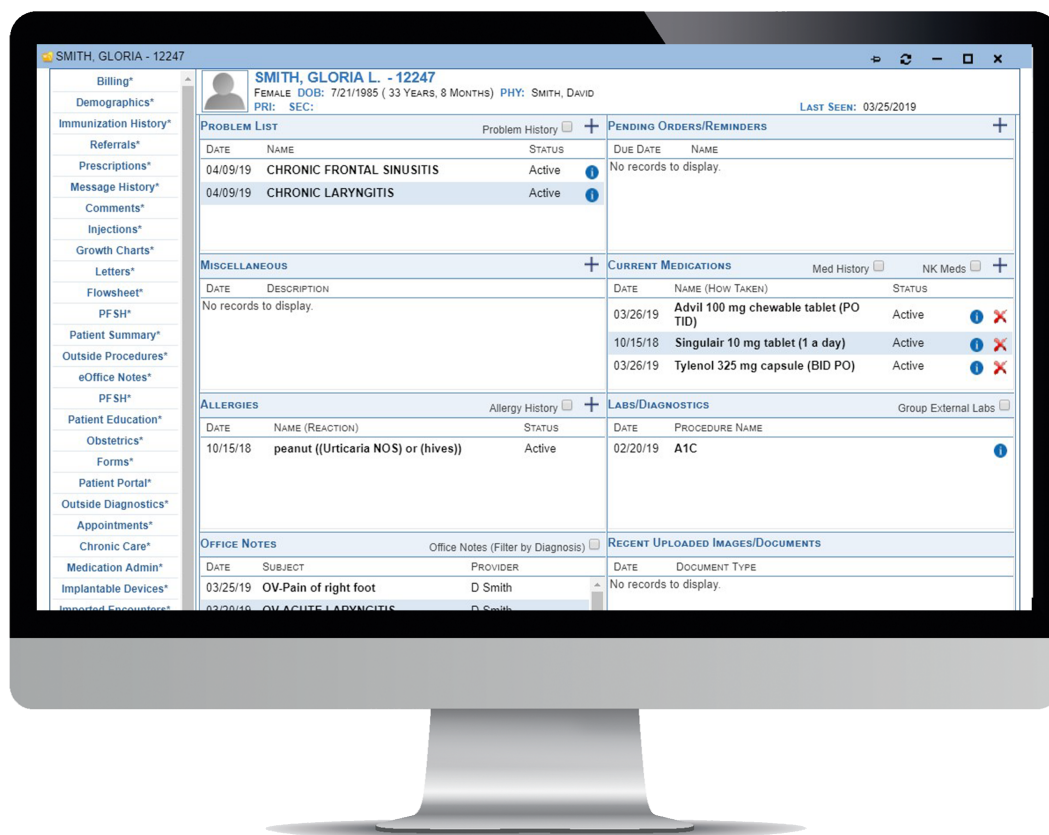


Figure 1.1: Sample image from the MedConnect Health Website [MedconnectInc \(2015\)](#) of their EHR system.

disruptive to their workflow [Tubaishat \(2018\)](#). Enhancing usability through intuitive interfaces, such as graph-based visualizations and natural language interfaces, may help mitigate this resistance.

Interoperability also remains an issue. Most EHR systems use proprietary data structures, making seamless data exchange between institutions difficult. A key advantage of using platforms like Neo4j in this study is the ability to visualize EHR data as interactive knowledge graphs that are more transferable and understandable across systems. For example, outputs can be exported in readable formats like PDF or JSON, preserving accessibility across environments.

Privacy and security are ongoing concerns. While this paper does not focus on cybersecurity directly, the integration of secure cloud services and robust access control mechanisms remains a prerequisite for deployment [Ahsan et al. \(2022\)](#).

Lastly, the concern that AI or EHR tools might replace human interaction is not applicable to the approach taken here. GRAG and LLM-based systems are intended as backend tools to support, not supplant, provider-patient relationships. These tools aim to reduce administrative burden, freeing up more time for human-centered care.

1.2 Towards an Intelligent EHR System

Despite the growing digitization of healthcare data, current EHR systems remain largely transactional and ill-equipped for advanced clinical reasoning or proactive decision support. To address this limitation, our work implements a hybrid architecture that blends structured medical data with the semantic understanding capabilities of large language models. Specifically, we integrate Graph Retrieval-Augmented Generation (GRAG) with vector-based similarity search to enable contextual, patient-specific question answering from graph-based EHRs.

At the heart of this system is a graph database (Neo4j), which stores patient-centric clinical records in a network of interconnected nodes—representing diagnoses,

procedures, and medications—along with timestamped edges capturing the chronology of care. This structured graph is extracted using Cypher queries and transformed into JSON-based patient narratives, sorted chronologically to maintain temporal fidelity.

To allow semantic querying, the extracted records are flattened into text blocks and embedded using the `thenlper/gte-large` transformer model via LangChain’s `HuggingFaceEmbeddings`. These embeddings are stored in a FAISS vector index, which enables fast and scalable similarity search. This approach empowers users to ask natural language questions like “*What treatment followed the patient’s ER visit?*”, with the system retrieving the most relevant events along with interpretable similarity scores.

This design aligns with recent advances in GRAG [Hu et al. \(2024\)](#), which combine symbolic knowledge representations (e.g., graphs) with neural generation models. Unlike traditional keyword search or rule-based systems, our vectorized framework allows nuanced retrieval by capturing semantic relationships between clinical events. Furthermore, the use of LangChain and FAISS allows for low-latency response times and high retrieval accuracy, making it feasible for integration into real-time clinical applications [Chase et al. \(2023\)](#); [Johnson et al. \(2017\)](#).

While our current deployment targets moderate-sized datasets, performance benchmarking (see Table 3.1) suggests strong potential for scalability. Future work may include embedding model distillation [Hinton et al. \(2015\)](#) or GPU-accelerated FAISS indexing [Johnson et al. \(2019\)](#) to support enterprise-scale data environments.

Ultimately, this pipeline lays the groundwork for intelligent EHR systems capable of answering complex temporal and semantic queries, offering clinicians meaningful insights without requiring deep technical expertise or manual chart review. The combination of graph-based structure and neural embeddings exemplifies a new paradigm in health informatics—one that brings reasoning closer to the point of care.

Chapter 2

Literature Review

2.1 Electronic Health Records

Electronic Health Records (EHRs) are digitized repositories of patient data accumulated across various clinical interactions. These records encompass a wide range of information including medical history, physician notes, vital signs, lab results, and prescriptions [Menachemi and Collum \(2011\)](#). Beyond resolving legibility concerns from handwritten notes [Rodríguez-Vera et al. \(2002\)](#), EHRs enable longitudinal tracking of patient health, support clinical decision-making, and facilitate data-driven research and public health monitoring [Jha et al. \(2009\)](#).

2.2 Comparison Between SQL and GRAG Systems

Structured Query Language (SQL) remains a dominant paradigm for querying structured data stored in relational databases [Melton and Simon \(2001\)](#). SQL databases are optimized for transactional workloads and operate efficiently when queries involve well-defined schemas and indexes. In contrast, GRAG systems combine semantic retrieval techniques with generative language models to provide

natural language responses to unstructured or semi-structured data [Lewis et al. \(2020\)](#). This section compares SQL and GRAG systems across three major dimensions: speed, flexibility, and cost.

2.2.1 Speed

SQL engines such as PostgreSQL and MySQL are designed for low-latency query execution. They employ mature indexing algorithms (e.g., B-trees, hash indexes) and query optimizers to return results in milliseconds for indexed data [Stonebraker and Rowe \(1991\)](#). In contrast, GRAG systems rely on embedding large volumes of data into vector space and performing similarity searches, which can introduce latency—especially if the embedding store is large or unoptimized. While modern vector databases like FAISS enables sub-second retrieval using techniques such as product quantization and approximate nearest neighbor (ANN) search, they are still typically slower than optimized SQL queries on small to moderate datasets [Johnson et al. \(2019\)](#).

Additionally, the generation step in GRAG adds further latency. Transformer-based models like those used in GRAG (e.g., LLaMA, GPT) require additional processing time to generate outputs, particularly if long-context understanding or reasoning is involved [Lewis et al. \(2020\)](#). However, this delay may be acceptable in applications where interpretability and semantic relevance outweigh the need for instant results.

2.2.2 Flexibility

SQL is rigid by design—it requires predefined schemas, and queries must conform to the structure of the underlying tables. Although powerful for structured data analysis, this limits its ability to query free-text, heterogeneous, or loosely structured data. Queries must often be rewritten or joined across multiple tables to extract complex relationships.

GRAG systems offer superior flexibility. By embedding documents, metadata, and natural language queries into a shared semantic space, GRAG supports natural language queries that capture intent and context without requiring knowledge of database schema. This is particularly valuable in domains like healthcare and legal research, where users may not be technically trained and data may span structured and unstructured formats [Guo et al. \(2023\)](#). The integration of graph-based document schemas further allows complex entity relationships (e.g., patient visit history, treatment plans) to be represented more naturally than in flat relational models [Yoon et al. \(2020\)](#).

2.2.3 Cost

Traditional SQL systems are cost-efficient for small-to-medium-scale deployments. Open-source engines (e.g., MySQL, PostgreSQL) have low setup and maintenance costs and can run efficiently on modest hardware. Their compute and memory requirements scale predictably with dataset size and query complexity.

GRAG systems, on the other hand, involve several cost-intensive components. Embedding documents using large language models (LLMs) incurs substantial upfront computational costs. Further, storing and querying high-dimensional embeddings requires specialized vector databases or GPU acceleration, which increase infrastructure expenses. Most significantly, the use of generative transformers for answer synthesis adds runtime inference costs—particularly when using commercial APIs or hosting large models on cloud platforms [Bommasani et al. \(2021\)](#).

That said, GRAG systems can reduce operational costs in settings where human-in-the-loop document retrieval or domain-specific search tasks are labor-intensive. The ability to answer open-ended questions semantically can streamline workflows that traditional SQL systems cannot easily support.

2.2.4 Summary

In summary, SQL excels in speed and cost-effectiveness for structured data scenarios, but is limited in flexibility and natural language support. GRAG systems offer greater adaptability and semantic understanding, particularly for unstructured or graph-based data, albeit at higher computational and financial costs. The choice between the two should be guided by the nature of the data, the expertise of end-users, and performance requirements.

2.3 LLMs

Before deploying a LLM, selecting a quantization strategy is critical. Quantization refers to the process of reducing the precision of model weights (e.g., from 32-bit to 8-bit) to reduce memory and computation requirements without significantly degrading performance [Gholami et al. \(2021\)](#). For this project, the available models included versions of Meta’s LLaMA 3 with 8 billion and 70 billion parameters [AI@Meta \(2024\)](#). While larger models typically demonstrate superior accuracy and reasoning capabilities, smaller models are often more efficient and environmentally sustainable [Polino et al. \(2018\)](#); [Frantar et al. \(2022\)](#).

In this study, we selected the 8-billion parameter model to simulate a realistic clinical deployment, balancing accuracy with resource efficiency. This model was deployed using the Viridian server cluster at the University of Tennessee, which comprises 238 CPU nodes and 6,584 cores [University of Tennessee \(nd\)](#).

SpaCy is a widely used industrial-strength NLP toolkit in Python that provides fast and accurate syntactic analysis of text. Its core features include tokenization, part-of-speech tagging, named entity recognition, and dependency parsing [Matthew Honnibal \(2022\)](#). Compared to tools like NLTK or Stanford’s Stanza, SpaCy offers superior speed and scalability, making it ideal for high-volume clinical text

preprocessing. Its extensibility and integration with transformer-based pipelines also enhance its utility in modern deep learning workflows.

2.4 GRAG

At the core of modern natural language processing (NLP) tasks lies the use of text embeddings, a technique that transforms natural language into high-dimensional vector representations. These vectors are designed to preserve semantic relationships such that texts with similar meanings reside closer together in vector space. This technique was popularized by Devlin et al. with the introduction of BERT (Bidirectional Encoder Representations from Transformers), which demonstrated that language models pre-trained on massive corpora could effectively capture contextual meaning across sentences and paragraphs [Devlin et al. \(2019\)](#).

Recent advancements in this area have led to the development of general-purpose embedding models such as GTE-large, released via Hugging Face by Li et al. [Li et al. \(2023\)](#). GTE-large is a sentence embedding model trained specifically for tasks like semantic search, question answering, and document classification. It is optimized to return dense, semantically meaningful vectors that generalize well across unseen queries.

After embedding text into this semantic space, it becomes possible to perform similarity search using distance metrics. While cosine similarity is commonly used in NLP tasks for comparing semantic similarity, the FAISS library (Facebook AI Similarity Search), used in this paper, defaults to the L2 distance (Euclidean distance) metric unless explicitly configured otherwise [Bishop \(2006\)](#). L2 distance measures the straight-line distance between vectors in high-dimensional space and is computed as:

$$\text{L2_distance}(\vec{A}, \vec{B}) = \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \quad (2.1)$$

This enables semantic retrieval, where relevant documents are retrieved based on meaning rather than literal keyword matches. To scale this operation efficiently over large datasets, researchers and engineers employ tools like FAISS (Facebook AI Similarity Search), an open-source library developed for fast similarity search and clustering of dense vectors [Johnson et al. \(2019\)](#). FAISS supports GPU acceleration and sophisticated indexing techniques such as inverted file systems (IVF) and product quantization (PQ), enabling it to operate efficiently even on billion-scale datasets.

A key part of embedding-based retrieval systems is the document schema—the structure that defines how content and metadata are stored. In frameworks like LangChain, documents are modeled with both page content and metadata, allowing for contextual retrieval that respects both the semantic meaning of a document and its structured attributes, such as patient identifiers or timestamps [Manning et al. \(2008\)](#). LangChain has emerged as a modular framework optimized for RAG-style applications, offering components that support embedding models, vector stores, document loading, and response synthesis [Chase et al. \(2023\)](#).

These methods form the foundation of retrieval-augmented generation (RAG), a framework in which retrieval systems are combined with generative models to produce accurate and context-aware responses to complex queries. Lewis et al. introduced this approach by combining a dense retriever with a generative transformer model, demonstrating superior performance on knowledge-intensive NLP tasks [Lewis et al. \(2020\)](#).

2.5 Data Visualization

One of the major shifts introduced by this method is the integration of a user interface that supports natural language queries. Unlike previous EHR systems that relied heavily on structured query forms, this interface allows users to interact with the system using free-form language, which improves accessibility and reduces friction in information retrieval. Errors in manual data entry can significantly skew research

outcomes [Barchard and Pace \(2011\)](#). By incorporating LLMs into the data entry process, semantic understanding of user intent can be leveraged to validate and structure inputs more accurately [Peng et al. \(2023\)](#). Therefore, this system offers a promising pathway to enhance both user experience and data integrity.

2.5.1 Graph Data Models in Healthcare

Graph-based data models offer a powerful and flexible framework for representing the complex, interconnected nature of clinical data. Unlike traditional relational databases, graph databases model relationships as first-class citizens, enabling more natural representation of medical ontologies, patient histories, and care pathways.

In a graph-based representation, entities such as patients, diagnoses, medications, and procedures are modeled as *nodes*, while relationships—such as “has diagnosis,” “was prescribed,” or “underwent procedure”—are modeled as *edges*. This structure is particularly effective for representing temporal sequences, comorbidities, and the complex web of factors influencing patient outcomes. For example, a patient node might be connected to multiple condition nodes (e.g., diabetes, hypertension) and medication nodes, allowing for a compact and semantically meaningful visualization of care trajectories.

2.5.2 Graph Databases and Semantic Querying

Graph databases such as Neo4j implement the *property graph* model, where nodes and edges can both carry key-value properties. This enables fine-grained querying, such as finding all patients over 60 with hypertension who were prescribed beta-blockers within a certain time window. Semantic querying languages like Cypher (used in Neo4j) or SPARQL (used with RDF graphs) support intuitive traversals of these relationships, making it easier to extract insights from otherwise sparse or unstructured records [Robinson et al. \(2015\)](#).

Furthermore, RDF-based representations, particularly those using HL7’s FHIR standard encoded in RDF, offer compatibility with semantic web technologies. This allows integration of external ontologies (e.g., SNOMED CT, LOINC) and supports automated reasoning over clinical data. The use of RDF and OWL enables inference engines to deduce new facts, such as identifying undiagnosed risk factors based on related comorbidities or medication history [HL7 Clinical Information Modeling Initiative \(2021\)](#).

2.5.3 Knowledge Graphs in Clinical Applications

Recent work has demonstrated the utility of domain-specific knowledge graphs (KGs) in healthcare. For example, PatientKG represents clinical events, diagnoses, and demographics as a comprehensive knowledge graph for cohort discovery, risk prediction, and temporal analysis [Zhang et al. \(2023\)](#). Such systems not only store data but also encode clinical logic and guidelines, bridging the gap between structured records and intelligent reasoning.

These graph-based models align naturally with RAG systems, enabling language models to ground their responses in structured, semantically rich representations. When integrated into a GRAG pipeline, graphs enhance both the precision of data retrieval and the interpretability of LLM-generated outputs, facilitating clinically meaningful decision support.

2.6 Clinical Decision Support Systems (CDSS)

CDSS are digital tools integrated with EHRs that provide clinicians with intelligently filtered knowledge to assist in decision-making. These systems range from basic alerts (e.g., drug-drug interaction warnings) to complex diagnostic assistance engines. CDSS have demonstrated improvements in clinical practice by reducing medication errors,

improving preventive service delivery, and enhancing diagnostic accuracy [Sutton et al. \(2020\)](#).

2.6.1 Limitations of Rule-Based Systems

Modern EHRs such as Epic and Cerner support integrated decision tools that leverage structured data (e.g., lab values, medication history) to trigger recommendations. However, the utility of these systems is often constrained by their reliance on rule-based logic, which lacks the flexibility to interpret nuanced patient scenarios. For instance, if-then rules cannot adapt to comorbidities or prioritize contradictory recommendations, limiting clinical value in complex cases.

Traditional CDSS approaches are typically static and rigid. They require manually curated knowledge bases, which are time-consuming to maintain and susceptible to obsolescence. Furthermore, clinicians often report "alert fatigue," where repeated warnings desensitize users and reduce system trust. As medical knowledge evolves rapidly, static systems struggle to accommodate new guidelines or rare edge cases without major reconfiguration.

2.6.2 LLMs for Dynamic Clinical Reasoning

LLMs offer a transformative alternative to traditional CDSS. By training on vast medical corpora and real-world case narratives, LLMs can generate context-aware responses, interpret ambiguous notes, and provide evidence-backed suggestions. For example, Meta's LLaMA 3 model can evaluate a patient's longitudinal history and generate plausible next steps, enhancing the diagnostic workflow. Deep learning architectures, when trained on large-scale health data, can rival specialist-level diagnostic performance in areas like dermatology and radiology [Esteva et al. \(2019\)](#).

In our GRAG framework, LLMs not only translate natural language queries into graph-based Cypher commands but also offer interpretive summaries, bridging symbolic reasoning with narrative clinical insight. This integration enables clinicians

to receive both structured search results and human-like justifications, effectively restoring the explanatory power lost in black-box AI systems.

Chapter 3

Methodology

Initially, data was pulled from a Kaggle database; however, the data culminated in a random forest algorithm that did not hold any strong indicative values. Examples for the previous data can be found in appendix [A.3](#). Some of the processes performed on these data points remain the same and are reflected in the current code. The data were changed over to real patient data, and therefore, the graphs have been redone.

Data used for this project is real patient data where the names and critical information of patients have been changed or removed to protect the anonymity of the patients. In the case where patient names are used for the examples shown in later sections, the names have been replaced by aliases. Other identifying information such as address and date of birth have been removed completely. They have no bearing on the outcome of this research and therefore are not included at all.

The full methodology of this project can be broken into Graph Query, Retrieval Augmented Generation, and Graph Retrieval Augmented Generation supported by Large Language Models. The outline of the methodology applied in a medical setting can be seen in Figure [3.1](#).

In order to create the GRAG and RAG, the data was read into a python script that pushed different categorizes dictionaries of information to the Neo4j database. The database consisted of a information panel that is shown in Figure [3.2](#).

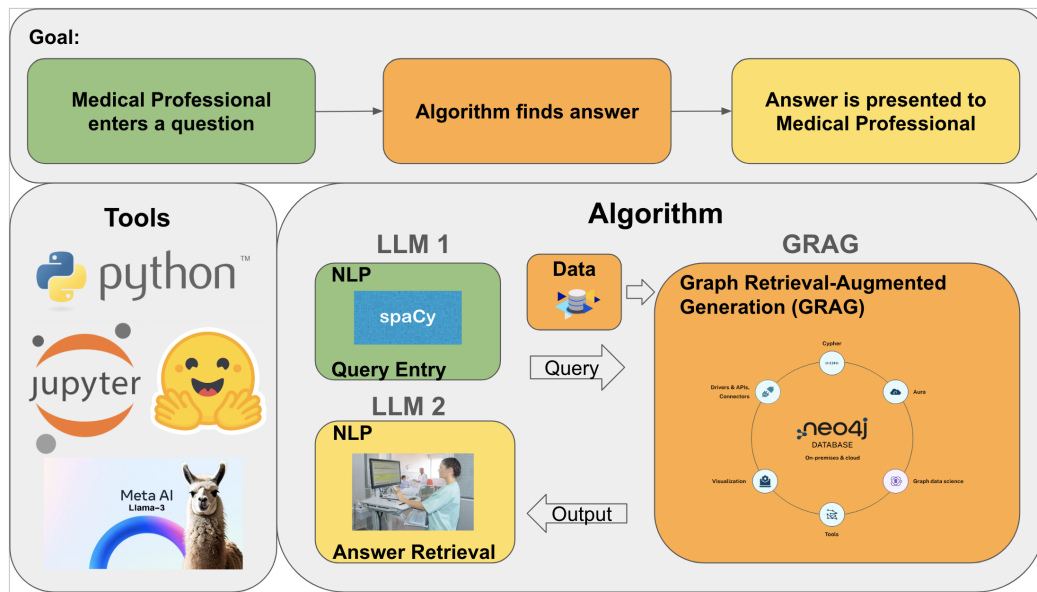


Figure 3.1: Hospital Workflow with GRAG and Graph Query Implementation

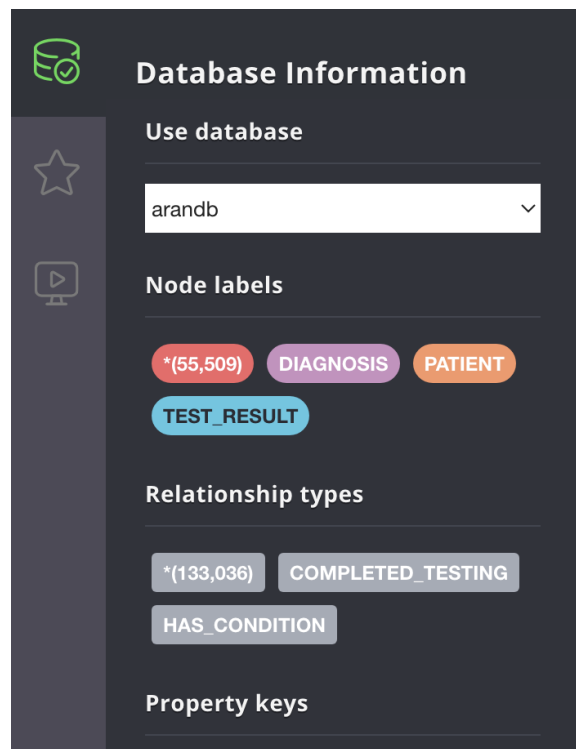


Figure 3.2: Neo4j database Node Types and Relationship Types

This panel gives a constant view of the Node and Relationship statuses, so that a user can check when additional information is added by checking for numerical changes.

3.1 Graph Query and Data Visualization

In order to initialize the Meta Llama 3 model, the necessary libraries needed to be installed. Beginning with the 'transformers' library, this library allows the connection between the Viridian Server and the Llama 3 website to download and install the selected model. The 8 billion parameter model consumes approximately 40 Gigabytes of GPU space to download [AI@Meta \(2024\)](#).

To analyze hospital admission patterns over time, we employed a Python-based system that integrates data retrieval from a Neo4j graph database, data transformation using `pandas`, and visualization using `matplotlib`. An interactive front-end interface was developed using `Gradio` to allow real-time graph generation. Images of the `gradio` frontend are available in the appendix [A.4](#) and [A.5](#).

3.1.1 Data Retrieval from Neo4j

We utilized the `neo4j` Python driver to establish a connection to a Neo4j graph database. The function `run_query()` accepts a Cypher query and returns the result as a `pandas` DataFrame. This is encapsulated in a try-except block to handle potential connection or query failures gracefully.

The `get_admission_data()` function executes a Cypher query to extract patients' admission dates from the Neo4j graph. The dates are then processed to derive the day of the week and the calendar week, which are essential for temporal trend analysis.

3.1.2 Graphical Visualization of Admission Trends

We implemented the `generate_bar_graph()` function to visualize hospital admission trends. It computes the average number of admissions for each day of the week across the available weeks. Additionally, it highlights the number of admissions on a given date (e.g., February 17, 2024) using a distinct color. This bar chart enhances interpretability by using rounded corners and clearly labeled axes.

For ease of use and demonstration, a Gradio web interface was developed. It consists of a button that, when clicked, calls the graph generation function and displays the output inline within a browser-based environment.

3.1.3 Summary

This methodology allows for dynamic querying, analysis, and visualization of hospital admission data using a graph-based backend. It supports better insight into weekly admission trends and facilitates real-time exploration through a user-friendly front-end interface.

3.2 RAG using Neo4j

3.2.1 Graph Data Extraction and Cleaning from Neo4j

This project utilizes Retrieval-Augmented Generation (RAG) and vector similarity search to enable semantic querying of structured electronic health records (EHRs). The methodology is composed of several sequential phases: data extraction, cleaning, temporal alignment, graph construction, vector embedding, and similarity search shown in Figure [3.3](#).

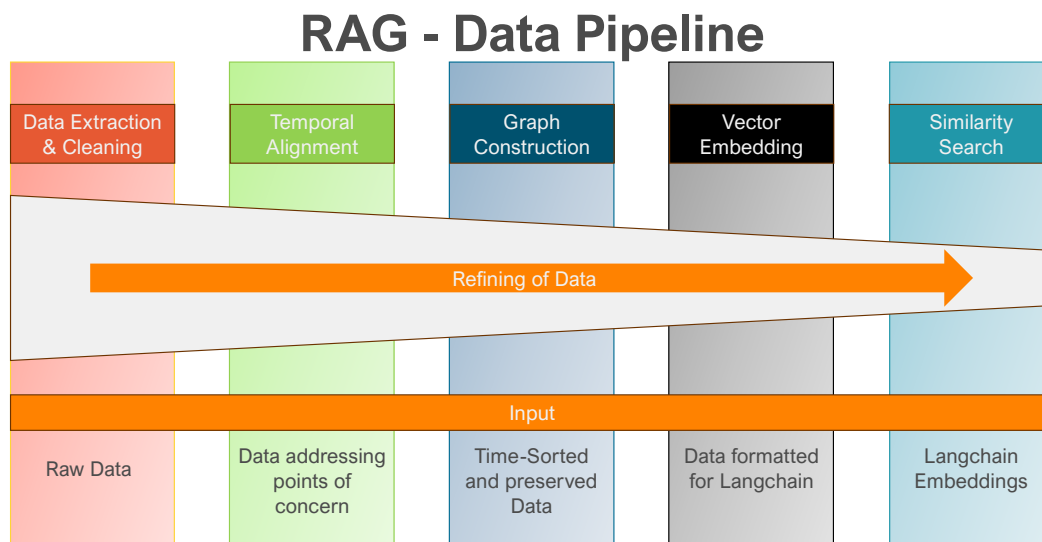


Figure 3.3: Graph-based data extraction and cleaning workflow from Neo4j to LangChain.

The patient data originates from a Neo4j graph database, where each patient node is connected to various diagnosis, medication, or procedure nodes via timestamped relationships. To see a sample of a patient node refer to the appendix [A.8.A](#) Cypher query is executed to retrieve all relationships and their properties, which are then processed into Python dictionaries.

- Null values and non-relevant fields (e.g., `index`, `count`) are removed.
- Redundant date/time fields are merged into a single unified timestamp format (e.g., `PERFORM_RESULT_DT_TM = PERFORM_RESULT_DT + PERFORM_RESULT_TM`).

- For simplicity of output, actions with multiple timestamps that occur incredibly close to one another and are almost redundantly creating 2 entries are using the first available timestamp.

3.2.2 Chronological Structuring and JSON Conversion

After extracting and cleaning, each patient's records are sorted chronologically. This allows the construction of a time-aligned narrative for each individual, which is later saved to a unified JSON file (`cancer_data_3.json`). The chronologically aligned records are shown in Figure 3.4.

- Each record becomes part of a sequential patient timeline.
- Pandas is used to create and sort dataframes per patient before re-serialization.

```
Patient data fetched successfully. Writing to file...
Sample data: {
  "1496928": [
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:47",
      "test_name": "Height in cm"
    },
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:47",
      "test_name": "Height"
    },
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:50",
      "test_name": "Temperature Temporal Artery"
    },
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:50",
      "test_name": "Systolic Blood Pressure"
    }
  ]
}
...
File size after writing: 140523554 characters.
Data successfully written to cancer_data_3.json.
```

Figure 3.4: Hierarchical JSON structure after patient-wise chronological alignment.

3.2.3 Embedding Patient Records into a Vector Space

To enable semantic search capabilities, the JSON data that has been structured in the above steps is converted into a format compatible with vector embeddings. Each patient's full record is flattened into a single text block per event, then embedded using the Hugging Face model "thenlper/gte-large". This process is shown in the workflow in Figure 3.5.

- Each record becomes a `langchain.Document` with metadata.
- LangChain's `HuggingFaceEmbeddings` is used to encode these records.
- A FAISS vector store is generated for fast similarity-based retrieval.

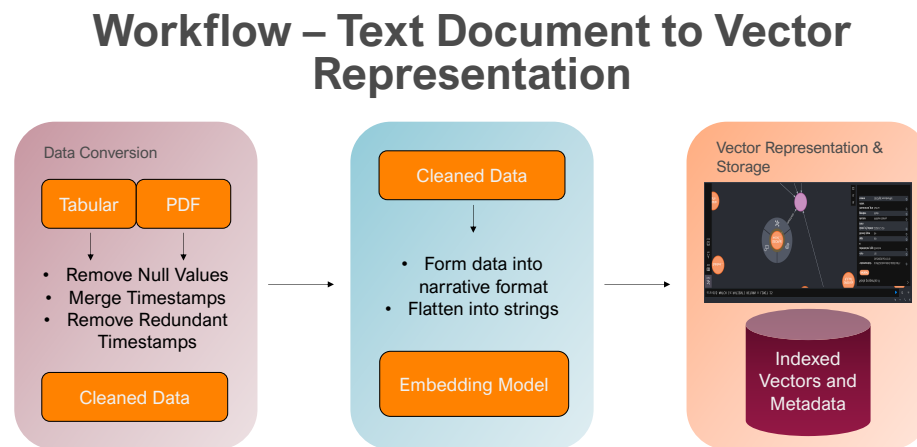


Figure 3.5: Workflow of transformation from text document to vector representation using LangChain and FAISS.

3.2.4 Semantic Search and Similarity Scoring

Once the FAISS index is built, semantic search queries (e.g., “*What is the next step for patient 358365 after ER DEPT VISIT?*”) are run against the vector database as shown in Figure 3.6.

- Top k most similar events are retrieved using cosine similarity.
- The results include both the matched record text and a similarity score, enabling interpretability and ranking.

3.2.5 Efficiency Considerations

While the FAISS-based system performs well for moderate-sized datasets, scaling to hundreds of thousands of records may require additional memory optimization or embedding model compression. The current approach balances semantic understanding and computational feasibility. Table 3.1 shows the comparison between multiple embedding models and the times they took as well as the accuracy of those models.

```
Vector store created successfully.
Performing similarity search for: What is the next step for patient 358365 after ER DEPT VISIT?

Matching Record:
Patient ID: 771398 | DX_DATE: 22-JAN-20 | index: 2146 | count: 4 | test_name: nan | admit_type: nan | procedure_code_desc: nan | dx_cd_desc:
c: Coma scale, best verbal response, oriented, at arrival to emergency department
Similarity Score: 0.9471635222434998

Matching Record:
Patient ID: 771398 | DX_DATE: 22-JAN-20 | index: 2122 | count: 4 | test_name: nan | admit_type: nan | procedure_code_desc: nan | dx_cd_desc:
c: Coma scale, best motor response, obeys commands, at arrival to emergency department
Similarity Score: 0.9721752405166626

Matching Record:
Patient ID: 868406 | PROCEDURE_DATE: 07-JUN-20 | index: 31 | count: 94 | dx_cd_desc: nan | procedure_code_desc: ER DEPT VST;EXP PRB FC H/
E;M | test_name: nan | admit_type: nan
Similarity Score: 0.9880189299583435
```

Figure 3.6: Visual representation of query-to-vector match with similarity score annotation.

Table 3.1: Performance Metrics for multiple Embedding Models

Model	thenlper/gte-large	sentence-transformers/all-MiniLM-L6-v2	thenlper/gte-base
Time Taken	1257.16 seconds	185.48 seconds	455.56 seconds
L2 Score	0.06	0.1812	0.0498
Top-k Accuracy	100%	60%	100%
Recall	1.00	1.00	1.00
Precision	1.00	0.33	1.00
Search Latency	0.6920 seconds	0.2565 seconds	0.5124 seconds

Due to the comparison of multiple models, one of the limitations of the research was the lack of use of parameterization.

3.2.6 Integration Summary

This methodology bridges structured EHR data with large language model capabilities by transforming graph-based data into a semantically searchable vector format. By combining Neo4j, LangChain, Hugging Face embeddings, and FAISS indexing, the pipeline allows for intelligent medical event retrieval with explainable similarity scoring.

3.3 GRAG

This study proposes a GRAG framework for enhancing clinical decision-making by integrating large language models with structured graph data from EHR. The system leverages Neo4j, a graph database, and Meta LLaMA 3, a local large language model served via Ollama, to provide interpretable and context-aware natural language responses to medical queries.

3.3.1 Graph Construction

We utilized a Neo4j graph database to model and store patient-related clinical information, including entities such as **PATIENT**, **DIAGNOSIS**, and relationships such as

HAS_DIAGNOSIS. Each patient is uniquely identified by a `patient.id`, and connected to diagnosis nodes through semantic relationships. The schema supports efficient retrieval via Cypher queries.

3.3.2 Graph Integration with LangChain

To interface the graph with a natural language querying system, we used the `Neo4jGraph` class from `langchain_community.graphs`, which enables secure and structured access to the Neo4j instance:

```
from langchain_community.graphs import Neo4jGraph

graph = Neo4jGraph(
    url="bolt://<HOST>:7687",
    username="<USERNAME>",
    password="<PASSWORD>",
    database="cancerdb",
    driver_config={"encrypted": False}
)
```

3.3.3 Language Model Configuration

To generate Cypher queries and produce human-readable answers, we used the Meta LLaMA 3 model via Ollama's local deployment interface. This avoids reliance on cloud-based APIs and ensures data privacy compliance:

```
from langchain_ollama import ChatOllama

llm = ChatOllama(model="llama3")
```

3.3.4 Graph-Aware Question Answering

We employed the `GraphCypherQAChain` from `LangChain`, which generates Cypher queries based on user input, executes the queries against the Neo4j graph, and summarizes the results:

```
from langchain.chains import GraphCypherQAChain

chain = GraphCypherQAChain.from_llm(
    llm=llm,
    graph=graph,
    verbose=True,
    allow_dangerous_requests=True
)
```

A user query such as “*What diagnoses are related to patient ID 1573975?*” is automatically translated into the following Cypher query:

```
MATCH (p:PATIENT {patient_id: <PATIENT_ID>})-[:HAS_DIAGNOSIS]->(d:DIAGNOSIS)
RETURN d.dx_cd_desc AS diagnosis;
```

3.3.5 Response Generation

The results of the Cypher query are retrieved from the graph and presented in a concise natural language format. For the example query above, the system outputs:

“Personal history of other drug therapy, Personal history of malignant neoplasm of breast, Primary osteoarthritis, right shoulder, Pain in right shoulder, Malignant neoplasm of upper-inner quadrant of right female breast, are related to patient ID 1573975.”

3.3.6 System Characteristics

The proposed GRAG system operates entirely offline, ensuring data privacy and HIPAA compliance. It provides an interpretable, explainable AI interface for medical professionals by combining symbolic reasoning (Cypher queries) with natural language generation capabilities.

Chapter 4

Results

4.1 Graph Query

Graph querying offers a dynamic approach to analyzing trends in hospital activity based on historical data. Rather than producing static answers, it provides visual insights akin to Google’s ”Popular Times” feature for businesses, which displays estimated peak hours based on aggregated user activity. An illustrative example of this visualization is shown in Figure 4.1.

Using this concept, a similar graph was generated based on historical hospital visitation data. This visualization aggregates and compares the average number of visits on a particular weekday with real-time values for a given date. As a proof of concept, Figure 4.2 demonstrates a version of this graph using legacy data outlined

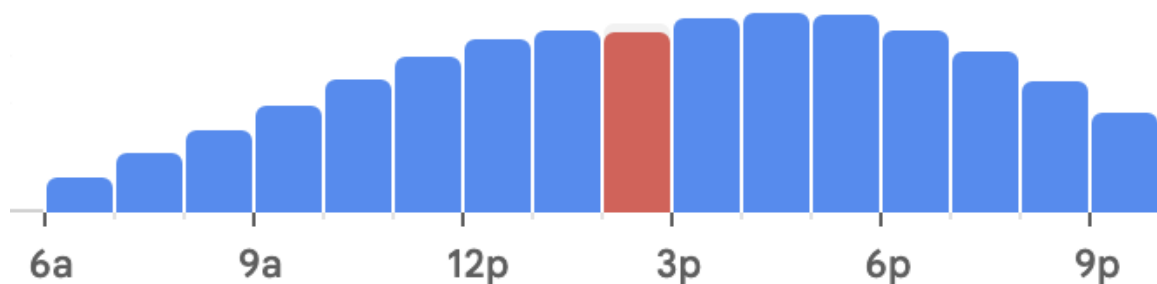


Figure 4.1: Google ’Busiest Times’ graph for Walmart [Alphabet \(2024\)](#).

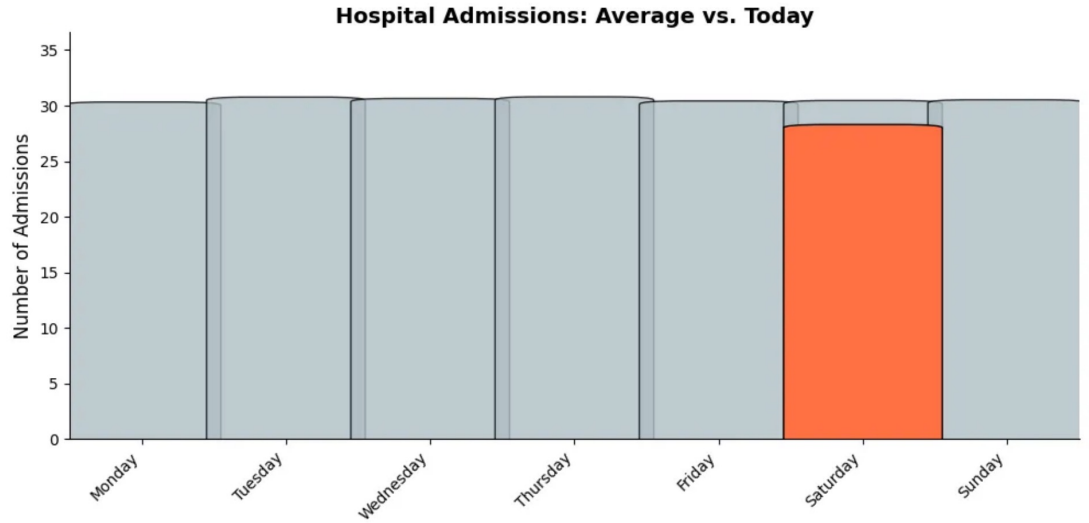


Figure 4.2: Hospital visitation trends by day of the week, compared to a specified date.

in the Methodology chapter. This representation can help administrators estimate expected traffic and plan staffing accordingly, offering valuable support for operational decision-making.

One of the key implementations of graph query functionality within our system is a predictive model designed to estimate a patient’s length of stay (LOS) in the hospital. This model utilizes a graph-based retrieval mechanism to identify similar historical patient cases and infer a likely duration of hospitalization for the current patient. Specifically, the system employs a similarity metric based on diagnostic profiles and other relevant clinical features to retrieve the 20 most closely matched patients from

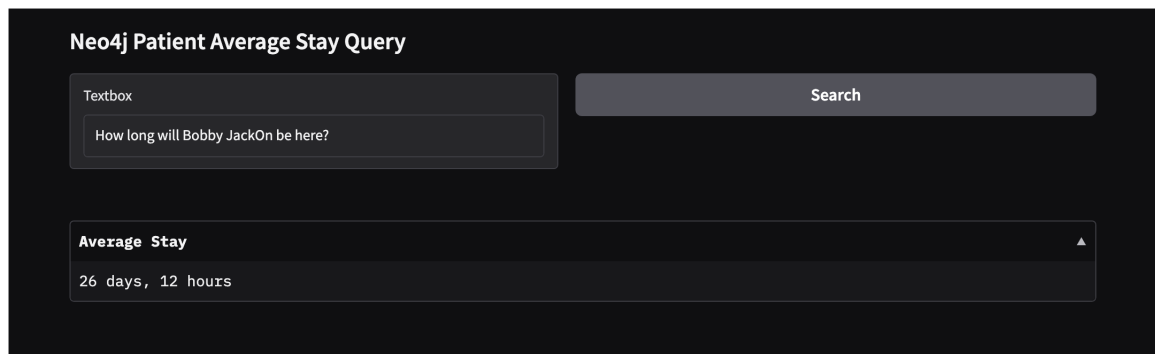
the existing graph database. The estimated LOS is then computed as the average duration of hospitalization among these similar cases.

To initiate this process, the user provides a natural language query that specifies the patient of interest. The system parses this query to extract the patient’s identity and associated clinical details. These details are used to query the graph, wherein nodes represent patients and edges encode relationships based on shared diagnoses, treatment regimens, or other medically significant attributes.

Figure 4.3 illustrates the outcome of such a prediction for the patient, Bobby Jackson. In this scenario, the user posed the query: *”How long will Bobby Jackson be here?”* The model responded by retrieving the 20 most recent and clinically similar cases to Bobby Jackson, calculating an average LOS of 26 days and 12 hours. This result offers an interpretable and data-driven estimate that can assist clinicians in resource planning and patient care strategies.

4.2 Example: Road to a Cure

To illustrate real-world use cases in a clinical setting, a series of test scenarios were developed. Scenario questions simulate typical queries from medical professionals seeking patient-specific information during various stages of diagnosis or treatment. This section outlines how the system responds to such inputs using retrieval-augmented generation (RAG) and graph-based querying (GRAG).



The screenshot displays a web interface titled "Neo4j Patient Average Stay Query". It features a "Textbox" containing the query "How long will Bobby Jackson be here?" and a "Search" button. Below the search area, a result box labeled "Average Stay" shows the output "26 days, 12 hours".

Figure 4.3: Graph-based estimated length of stay for patient Bobby Jackson.

4.2.1 Scenario

To effectively test the methodology, a patient with comprehensive medical records, including diagnoses, prescriptions, lab results, and physical tests, was selected. The patient, pseudonymously referred to as *Jennifer Anthony*, has a well-documented medical history within the Neo4j graph database. Her node, along with all related clinical interactions, is depicted in Figure 4.4.

Example queries posed in this scenario include:

- What treatments or procedures has Jennifer Anthony undergone prior to her current appointment?
- What medications has she been prescribed or taken historically?

These questions aim to simulate common queries that a clinician might make during patient intake or treatment review. To further evaluate the system’s performance, structured data inputs—such as a recorded temperature reading—were also submitted to the query generator for transformation and insertion into the database.

4.2.2 System Use

To explore the patient’s drug history, the RAG system was queried using the prompt: `*"When did Patient ID: 630439 take DENOSUMAB (Prolia) 60 mg/mL SYRINGE?"*` The system returned the relevant clinical event, as shown in Figure 4.5, effectively pinpointing the date and details of the administration.

A broader query—`*"What drugs are related to patient ID 630439?"*`—was submitted to the GRAG system. The graph-based query returned all associated medications, with results consistent with the Neo4j data for Jennifer Anthony (Figure 4.6). This not only validates the integrity of the graph-based method but also highlights its capacity to return comprehensive and accurate medical histories efficiently.

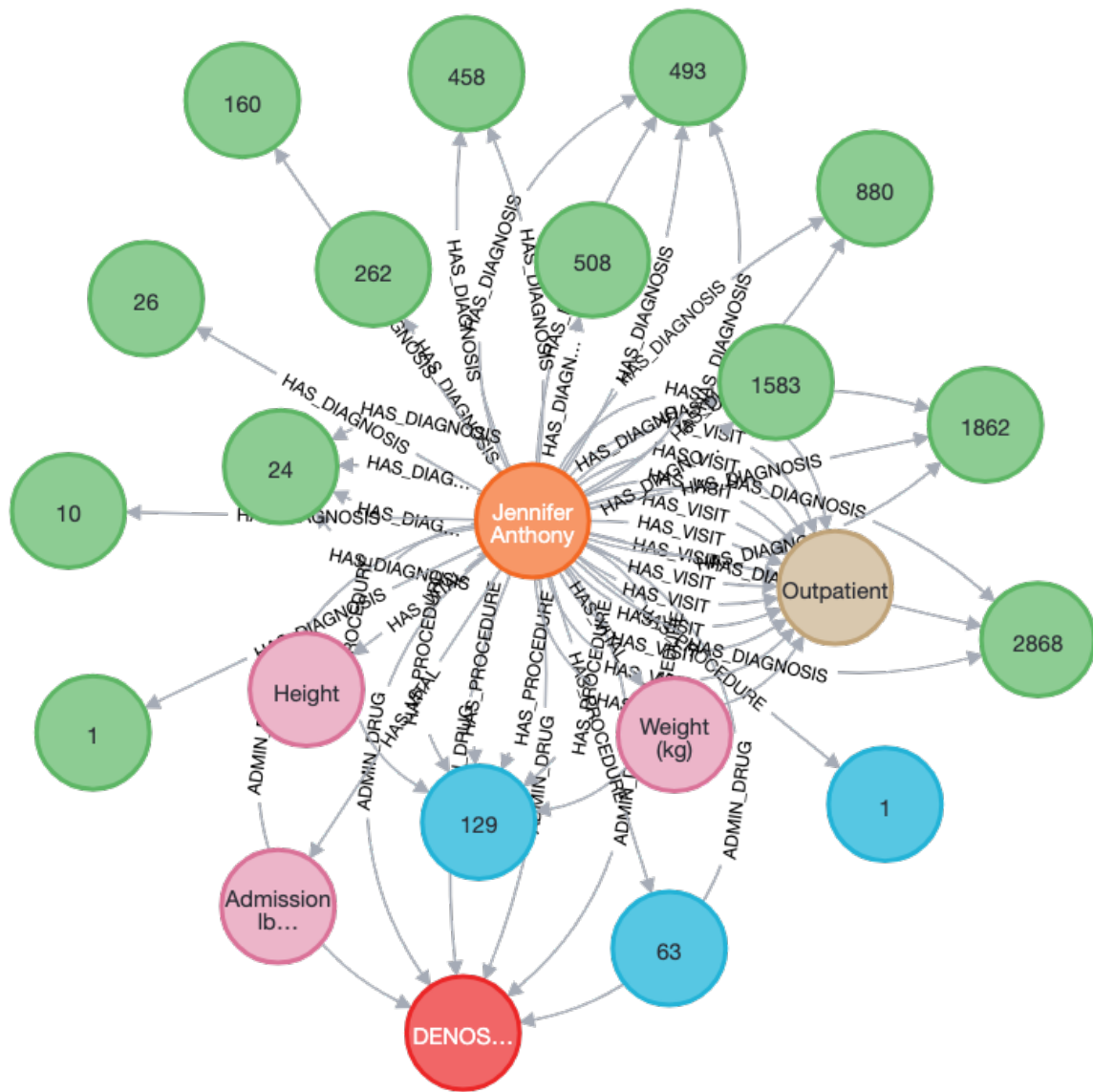


Figure 4.4: Expanded Neo4j node for Jennifer Anthony, displaying interconnected nodes for diagnoses, lab tests, medications, and other clinical events.

Performing similarity search for: When did Patient ID: 630439 take DENOSUMAB (Prolia) 60 mg/mL SYRINGE?

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 27-OCT-23 10:00.

Similarity Score: 0.059966202825307846

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 26-APR-23 10:00.

Similarity Score: 0.06096201390028

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 21-APR-22 10:15.

Similarity Score: 0.0609792098402977

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 21-APR-21 16:00.

Similarity Score: 0.060987427830696106

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 21-OCT-21 11:15.

Similarity Score: 0.06163030117750168

Matching Record:

Patient ID: 630439 | narrative: Patient 630439 had a test for DENOSUMAB (Prolia) 60 mg/mL SYRINGE on 21-OCT-22 08:30.

Similarity Score: 0.06239713355898857

Figure 4.5: RAG output for the query: "When did Patient ID: 630439 take DENOSUMAB (Prolia) 60 mg/mL SYRINGE?"

```
> Entering new GraphCypherQAChain chain...

Generated Cypher:

MATCH (p:PATIENT {patient_id: 1198337})-[:ADMIN_DRUG]->(d:DRUG) RETURN
d.drug_name AS drugName;
Full Context
[{'drugName': 'ceFAZolin'}, {'drugName': 'DENOSUMAB (Prolia) 60 mg/mL SYRINGE'},
{'drugName': 'DENOSUMAB (Prolia) 60 mg/mL SYRINGE'}, {'drugName': 'DENOSUMAB
(Prolia) 60 mg/mL SYRINGE'}, {'drugName': 'DENOSUMAB (Prolia) 60 mg/mL SYRINGE'},
{'drugName': 'acetaminophen'}, {'drugName': 'ceFAZolin in D5W'},
{'drugName': 'acetaminophen'}, {'drugName': 'D5 1/2 NS-2@ mEq Potas Chlor 20 mEq'},
{'drugName': 'D5 1/2 NS-20 mEq Potas Chlor 20 mEq'}]

> Finished chain.
ceFAZolin, DENOSUMAB (Prolia) 60 mg/mL SYRINGE, acetaminophen.
```

Figure 4.6: GRAG output for the query: "What drugs are related to patient ID 630439?"

To further validate system scalability, the same drug-querying method was applied to a different patient with a more extensive medication history. Figure 4.7 displays the output for Patient ID: 1573975, showing multiple drug associations returned accurately by the GRAG system.

To simulate data entry workflows, the query generation system was tested using an input representing a temperature reading. The input, "Create Temporal Artery Temperature, where RESULT: 97.4, for patient ID, 358345, on date, PERFORM RESULT DT: 26-AUG-24", was processed and transformed into a database-ready Cypher query (Figure 4.8).

This output illustrates the system's ability to correctly format and categorize clinical data for insertion, even when multiple types of temperature readings exist. Notably, the query is executable immediately upon generation, enabling seamless integration into clinical databases without requiring additional manual adjustments.

> Entering new GraphCypherQAChain chain...

Generated Cypher:

```
MATCH (p:PATIENT {patient_id: 1573975})-[:HAS_DIAGNOSIS]->(d:DIAGNOSIS) RETURN
d.dx_cd_desc AS diagnosis;
```

Full Context:

```
[{'diagnosis': 'Personal history of other drug therapy'}, {"diagnosis": 'Personal history of
malignant neoplasm of breast'}, {'diagnosis': 'Other specified postprocedural states'},
{'diagnosis': 'Encounter for screening mammogram for malignant neoplasm of breast'},
{'diagnosis': 'Encounter for other administrative examinations'}, {'diagnosis': 'Primary
osteoarthritis, right shoulder'}, {'diagnosis': 'Pain in right shoulder'}, {'diagnosis': 'Malignant
neoplasm of upper-inner quadrant of right female breast'}, {'diagnosis': 'Personal history of
malignant neoplasm of breast'}, {'diagnosis': 'Encounter for screening mammogram for
malignant neoplasm of breast'}]
```

> Finished chain.

Personal history of other drug therapy, Personal history of malignant neoplasm of breast, Primary osteoarthritis, right shoulder, Pain in right shoulder, Malignant neoplasm of upper-inner quadrant of right female breast, are related to patient ID 1573975.

Figure 4.7: GRAG output for the query: "What drugs are related to patient ID 1573975?"

```

User input: Create Temporal Artery Temperature, where RESULT: 97.4, for patient ID, <id>:
358345, on date, PERFORM_RESULT_DT: 26-AUG-24
Generated Cypher Query:
MATCH (p:PATIENT {patient_id: toLower("358345")})
WHERE p.vital_status = 'alive'
WITH p AS patient
MATCH (v:VITAL {ADMISSION_ID: p.patient_id, PERFORM_RESULT_DT: datetime({date:
"26—Aug-24", time: "00:00:00"})})
WHERE v.RESULT = 97.4 AND v RESULT STATUS = 'normal'
RETURN patient.name AS name, v.TEST_CD AS test_cd
Error: {code: Neo.ClientError.Statement.SyntaxError} {message: Invalid input
'RESULT_STATUS': expected an expression, 'FOREACH', 'ORDER BY', 'CALL', 'CREATE',
'LOAD CSV', 'DELETE', 'DETACH', 'FINISH', 'INSERT', 'LIMIT', 'MATCH', 'MERGE',
'NODETACH', 'OFFSET', 'OPTIONAL', 'REMOVE', 'RETURN', 'SET', 'SKIP', 'UNION', 'UNWIND',
'USE', 'WITH' or <EOF> (line 5, column 29 (offset: 240)) "
WHERE v.RESULT = 97.4 AND v RESULT_STATUS = 'normal'"

```

Figure 4.8: Generated Cypher query for inserting a temporal artery temperature reading into the Neo4j database.

Chapter 5

Conclusions and Future Work

5.1 Conclusion

This study demonstrates the promising capabilities of RAG, GRAG, and LLMs for enhancing data visualization, semantic querying, and automated data entry in healthcare settings. By integrating these technologies with graph databases such as Neo4j, we illustrate a novel approach to interacting with EHRs in a more intuitive, dynamic, and user-centered manner.

Our findings suggest that while traditional database systems like SQL, EPIC, and Cerner remain highly efficient for structured data retrieval, they lack the flexibility and interpretability that modern LLM-based solutions offer. GRAG and LLMs, though sometimes slower in raw computational speed, offer a more intuitive and context-aware interface that may streamline workflows for clinicians and researchers by reducing the cognitive burden associated with complex query languages.

The case study further highlights these advantages: the models performed effective lookups similar to SQL, but extended functionality by offering contextual outputs, visualizations (e.g., Figure 4.2), and even predictive capabilities. Notably, GRAG's ability to infer potential future patient conditions or hospital traffic based on past

graph-based trends represents a unique and impactful extension beyond traditional systems, none of which currently incorporate such predictive insight natively.

5.2 Future Work

One of the primary challenges encountered in this project was bridging the gap between industrial engineering methodologies and healthcare informatics. While the technical deployment of graph-based systems and LLMs was facilitated by accessible learning resources, translating these insights into clinically relevant applications remains a substantial hurdle. A vital next step involves interdisciplinary collaboration, particularly with clinicians, medical researchers, and healthcare students who can assess the system’s practical utility and help align its outputs with real-world clinical reasoning.

Regarding predictive applications, one promising but underdeveloped area is the estimation of patient length of stay (LOS). Our system hints at the potential to use graph-based data, such as diagnosis severity, treatment regimens, and medication effects, to formulate LOS predictions. However, this remains a complex, multifactorial challenge that warrants a dedicated research initiative. Future work should focus on developing a robust, explainable model that leverages graph structures to correlate clinical variables with stay durations—potentially enabling better resource allocation, bed management, and discharge planning.

Another important future direction involves infrastructure development. For this system to be viable in real clinical environments, additional middleware and scripting layers are necessary to integrate these query generation and data entry mechanisms into existing healthcare IT frameworks. This includes:

- Automated transformation of natural language inputs into executable database commands
- Real-time validation and error-checking of inputs

- Support for custom workflows, edge cases, and clinician-specific preferences
- Consideration of Human-Computer Interaction (HCI) principles to enhance usability, reduce friction, and ensure compliance with healthcare regulations and ergonomics

Additional improvements that could be made to the current model based on feedback include:

- Automated tracking of changes made by the current user for a shift change feature
- Parameterization of the large language model and incorporation of a larger sample set for testing
- Better integration of commonly used questions by doctors
- Integration with digital twin systems for seamless training programs for medical professionals
- More user friendly front-end solution

Ultimately, the true value of this approach lies in its adaptability and extensibility. With further refinement and collaboration between engineers and healthcare professionals, these intelligent systems can play a critical role in improving decision support, reducing documentation burdens, and unlocking insights from complex, multimodal medical data.

Bibliography

- Ahsan, M., Nygard, K. E., Gomes, R., Chowdhury, M. M., Rifat, N., and Connolly, J. F. (2022). Cybersecurity threats and their mitigation approaches using machine learning—a review. *Journal of Cybersecurity and Privacy*, 2(3):527–555. [4](#)
- AI@Meta (2024). Llama 3 model card. [9](#), [18](#)
- Alphabet (2024). Google search result. [Accessed 10-06-2024]. [x](#), [28](#)
- Barchard, K. A. and Pace, L. A. (2011). Preventing human error: The impact of data entry methods on data accuracy and statistical results. *Computers in Human Behavior*, 27(5):1834–1839. [12](#)
- Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer. [10](#)
- Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. (2021). On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*. [8](#)
- Chase, H. et al. (2023). Langchain: Building applications with llms through composability. <https://www.langchain.com>. [5](#), [11](#)
- Cunningham, R. (2009). Stimulus bill implementation: Expanding meaningful use of health IT. *Issue Brief George Wash. Univ. Natl. Health Policy Forum*, (834):1–16. [2](#)

- Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (long and short papers)*, pages 4171–4186. 10
- Esteva, A., Robicquet, A., Ramsundar, B., Kuleshov, V., DePristo, M., Chou, K., Cui, C., Corrado, G., Thrun, S., and Dean, J. (2019). A guide to deep learning in healthcare. *Nature Medicine*, 25(1):24–29. 14
- Feinstein, J. S. (1993). The relationship between socioeconomic status and health: a review of the literature. *The Milbank Quarterly*, pages 279–322. 3
- Frantar, E., Stock, P., and Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pretrained transformers. *arXiv preprint arXiv:2210.17323*. 9
- Gholami, A., Kim, S., Yao, Z., Mahoney, M., and Keutzer, K. (2021). A survey on quantization of deep neural networks. *arXiv preprint arXiv:2103.13630*. 9
- Griebel, L., Prokosch, H.-U., Köpcke, F., Toddenroth, D., Christoph, J., Leb, I., Engel, I., and Sedlmayr, M. (2015). A scoping review of cloud computing in healthcare. *BMC Medical Informatics and Decision Making*, 15:1–16. 1
- Guo, Y., Li, Z., Jin, X., Liu, Y., Zeng, Y., Liu, W., Li, X., Yang, P., Bai, L., Guo, J., and Cheng, X. (2023). Retrieval-augmented code generation for universal information extraction. *arXiv preprint arXiv:2311.02962*. 8
- Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*. 5
- HL7 Clinical Information Modeling Initiative (2021). Fhir rdf: Rdf for hl7 fhir. <https://www.hl7.org/fhir/rdf.html>. Accessed: 2025-04-22. 13

- Hu, Y., Lei, Z., Zhang, Z., Pan, B., Ling, C., and Zhao, L. (2024). Grag: Graph retrieval-augmented generation. *arXiv preprint arXiv:2405.16506*. 5
- Jha, A. K., DesRoches, C. M., Campbell, E. G., Donelan, K., Rao, S. R., Ferris, T. G., Shields, A., Rosenbaum, S., and Blumenthal, D. (2009). Use of electronic health records in us hospitals. *New England Journal of Medicine*, 360(16):1628–1638. 6
- Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with gpus. <https://github.com/facebookresearch/faiss>. Accessed: 2025-04-23. 5
- Johnson, J., Douze, M., and Jégou, H. (2019). Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547. 5, 7, 11
- Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Kuttler, H., Lewis, M., tau Yih, W., Rocktäschel, T., and Riedel, S. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474. 7, 11
- Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., and Zhang, M. (2023). Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*. 10
- Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press. 11
- Matthew Honnibal, I. M. (2022). 9
- McGinn, C. A., Grenier, S., Duplantie, J., Shaw, N., Sicotte, C., Mathieu, L., Leduc, Y., Légaré, F., and Gagnon, M.-P. (2011). Comparison of user groups’ perspectives of barriers and facilitators to implementing electronic health records: a systematic review. *BMC Medicine*, 9:1–10. 2, 3
- MedconnectInc (2015). Medconnecthealth EHR System. x, 3

- Melton, J. and Simon, A. R. (2001). *Understanding the New SQL: A Complete Guide*. Morgan Kaufmann. [6](#)
- Menachemi, N. and Collum, T. H. (2011). Benefits and drawbacks of electronic health record systems. *Risk Management and Healthcare Policy*, pages 47–55. [6](#)
- Nick Minaie, P. (2024). Are small language models (slm) the future of llms? [3](#)
- Peng, Y., Shen, Y., Zhang, Y., and Zhang, Y. (2023). Towards robust medical text summarization: A new dataset and decomposition method. *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*. [12](#)
- Polino, A., Pascanu, R., and Alistarh, D. (2018). Model compression via distillation and quantization. *arXiv preprint arXiv:1802.05668*. [9](#)
- Robinson, I., Webber, J., and Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data*. O’Reilly Media, Inc., 2nd edition. [12](#)
- Rodríguez-Vera, F. J., Marin, Y., Sanchez, A., Borrachero, C., and Pujol, E. (2002). Illegible handwriting in medical records. *Journal of the Royal Society of Medicine*, 95(11):545–546. [6](#)
- Stonebraker, M. and Rowe, L. A. (1991). The postgres next-generation database management system. *Communications of the ACM*, 34(10):78–92. [7](#)
- Sultan, N. (2014). Making use of cloud computing for healthcare provision: Opportunities and challenges. *International Journal of Information Management*, 34(2):177–184. [1](#)
- Sutton, R. T., Pincock, D., Baumgart, D. C., Sadowski, D. C., Fedorak, R. N., and Kroeker, K. W. (2020). An overview of clinical decision support systems: benefits, risks, and strategies for success. *NPJ Digital Medicine*, 3(1):17. [14](#)

- Tubaishat, A. (2018). Perceived usefulness and perceived ease of use of electronic health records among nurses: Application of technology acceptance model. *Informatics for Health and Social Care*, 43(4):379–389. 3, 4
- University of Tennessee (n.d.). System Overview on ISAAC Legacy — Office of Innovative Technologies — oit.utk.edu. <https://oit.utk.edu/hpsc/isaac-open/open-enclave-system-overview/>. [Accessed 10-06-2024]. 9
- Wang, L. (2019). From intelligence science to intelligent manufacturing. engineering, 5 (4), 615–618. 1
- Yoon, B.-H., Kim, S.-K., and Kim, S.-Y. (2020). Graph-representation of patient data: A systematic literature review. *Journal of Medical Systems*, 44(6):1–10. 8
- Zhang, Y., Zhou, J., Wang, Y., Zhang, R., and Xu, H. (2023). Patientkg: A knowledge graph for representing and analyzing patient data. *Journal of Biomedical Informatics*, 136:104300. 13

Appendix A

Experimental Results

A.1 Images - Old Data - Correlation Matrix and Random Forest Results

Random Forest Metrics:

	Output Variable	R-squared	MSE
0	Medical Condition	0.060158	2.760566
1	Test Results	0.064902	0.630802

Figure A.1: Random Forest Algorithm R-squared and mean squared error

Random Forest Metrics:						
	Output Variable	R-squared	MSE	MAE	RMSLE	RMSE
0	Medical Condition	-0.014043	2.978513	1.442402	0.390494	1.725837
1	Test Results	-0.014652	0.684468	0.698661	0.215953	0.827326

Figure A.2: Random Forest Algorithm R-squared, Mean Squared Error, Mean Absolute Error, Root Mean Squared Log Error, and Root Mean Squared Error

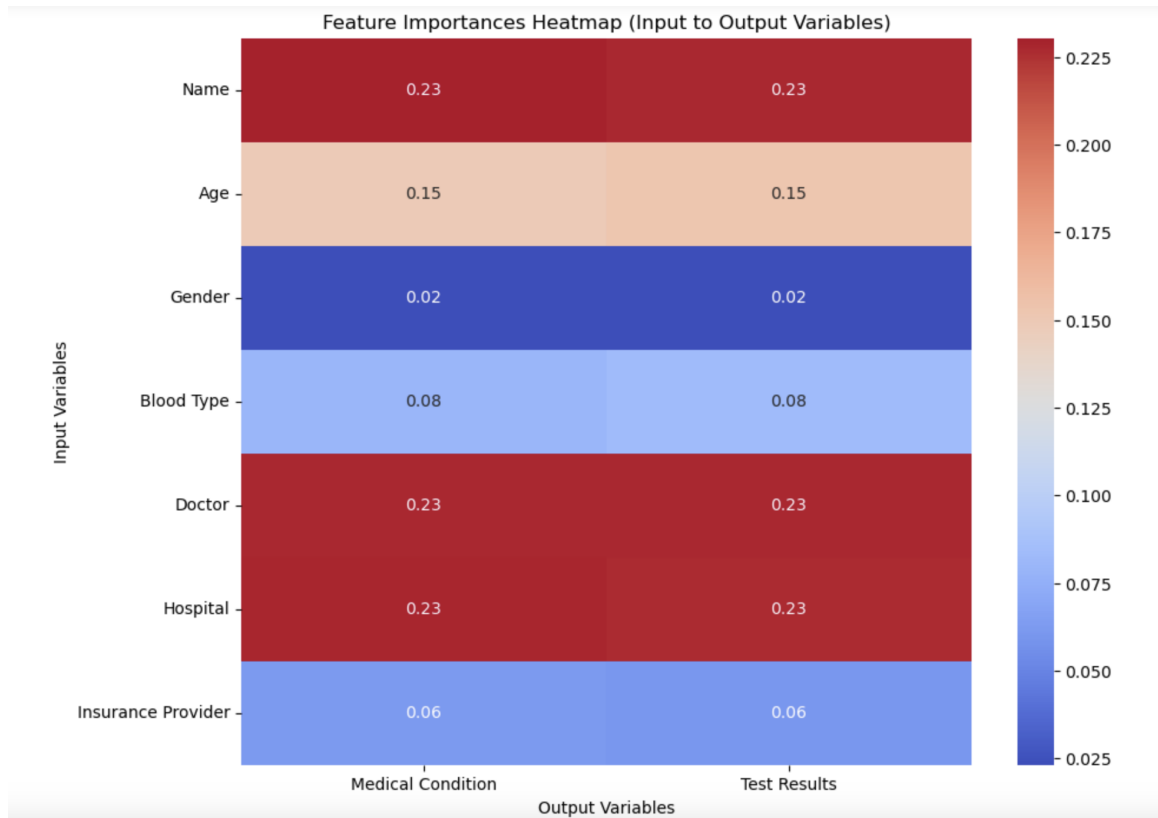


Figure A.3: Correlation Matrix for Random Forest Algorithm Generated for Healthcare Data

A.2 Images - Old Data - Neo4j Sample Queries in Gradio display

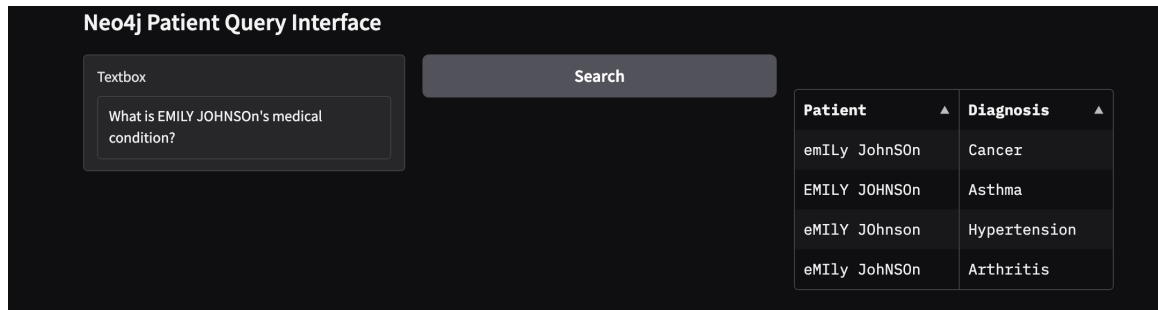


Figure A.4: Sample of the model extracting information on all patients named Emily Johnson

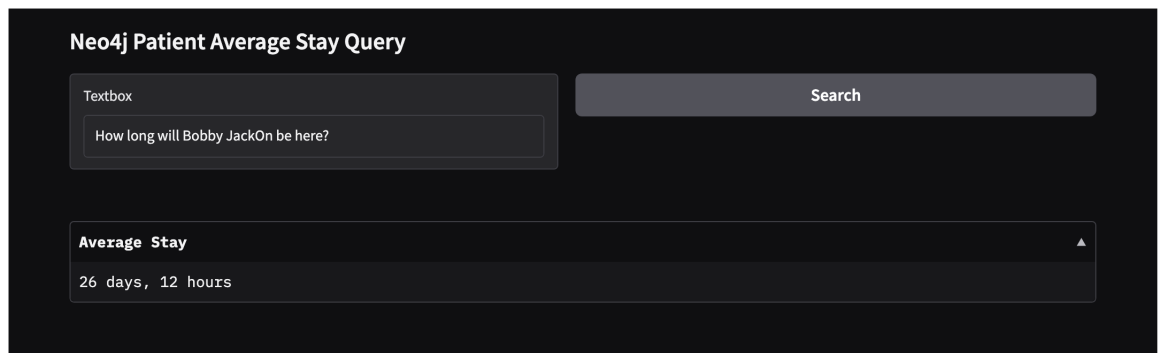


Figure A.5: Average stay time for Patient, Bobby JacksOn, taken from an average of the 20 most similar patients determined by the model

A.3 Images - Previous JSON Structures

First 5000 characters of the file:

```
{
  "1496928": [
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:47",
      "DATE_TIME_OF_ADMISSION": null,
      "DATE_TIME_OF_DISCHARGE": null,
      "START_DT_TM": null,
      "ADMIN_DT_TM": null,
      "ORDER_DT_TM": null,
      "DX_DATE": null,
      "PROCEDURE_DATE": null,
      "index": 24,
      "count": 3127,
      "dx_cd_desc": NaN,
      "procedure_code_desc": NaN,
      "drug_name": NaN,
      "admit_type": NaN,
      "test_name": "Height in cm"
    },
    {
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:47",
      "DATE_TIME_OF_ADMISSION": null,
      "DATE_TIME_OF_DISCHARGE": null,
      "START_DT_TM": null,
      "ADMIN_DT_TM": null,
      "ORDER_DT_TM": null,
      "DX_DATE": null,
      "PROCEDURE_DATE": null,
      "index": 16,
      "count": 7060,
      "dx_cd_desc": NaN,
      "procedure_code_desc": NaN,
      "drug_name": NaN,
      "admit_type": NaN,
      "test_name": "Height"
    },
  ],
}
```

Figure A.6: Initial JSON structure

```

Patient data fetched successfully. Writing to file...
Sample data: {
  "1496928": [
    {
      "DX_DATE": NaN,
      "dx_cd_desc": NaN,
      "PROCEDURE_DATE": NaN,
      "procedure_code_desc": NaN,
      "START_DT_TM": NaN,
      "ADMIN_DT_TM": NaN,
      "drug_name": NaN,
      "DATE_TIME_OF_ADMISSION": NaN,
      "DATE_TIME_OF_DISCHARGE": NaN,
      "admit_type": NaN,
      "PERFORM_RESULT_DT_TM": "25-JUL-23 12:47",
      "test_name": "Height in cm"
    },
    {
      ...
  ]
}
File size after writing: 563118911 characters.
Data successfully written to cancer_data_2.json.

```

Figure A.7: JSON structure 2nd attempt

A.4 Image - Neo4j - Patient Node and Diagnosis Node

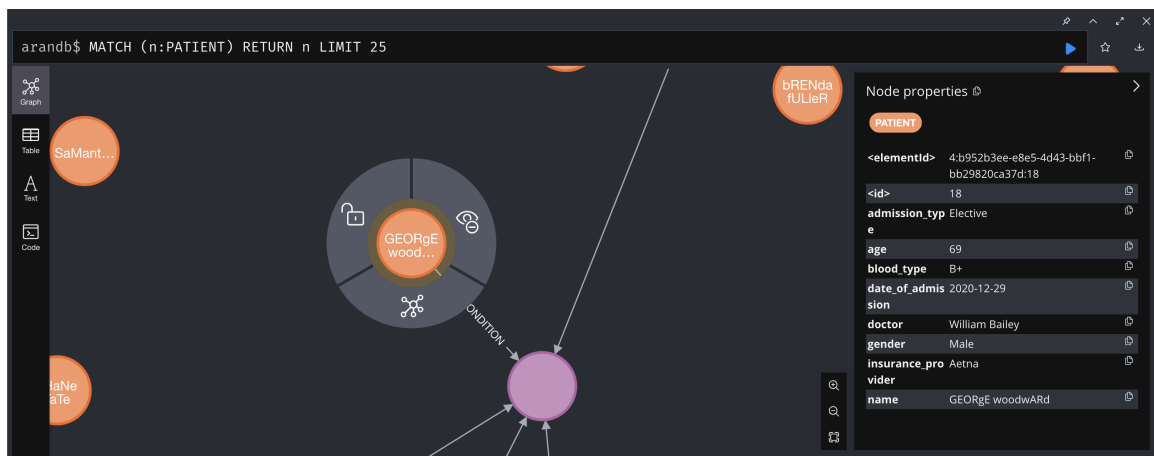


Figure A.8: Patient node, George Woodward, in neo4j GRAG database

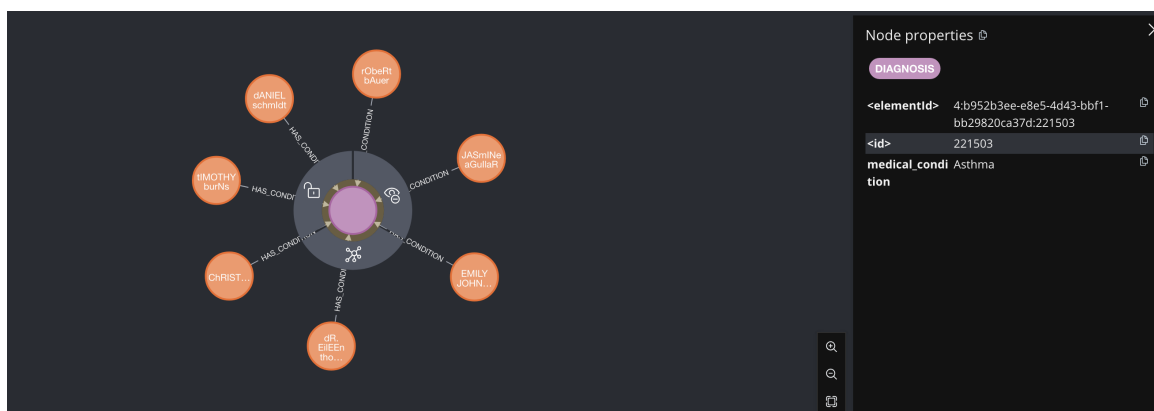


Figure A.9: Diagnosis node, Asthma, in neo4j GRAG database

Vita

The author of this work is Seyed Aran Arab. Born in Knoxville, Tennessee, Aran is a 26 master's student seeking to immerse himself in as much knowledge as the University of Tennessee's resources would allow. Alongside that goal, a curiosity with modern technology and its applications across multiple fields led him to the pursuit of a Master's Degree.

Going forward, Aran is looking to pursue success in industry by starting a business surrounding a niche application of Large Language Models or join a company with low technological constraints. This interest stems from his time in his assistant-ship. The experience shaped his outlook on the broad and unexplored applications of modern technology such as GRAG, LLMs, and NLP in a multitude of fields.