Explainable HybridRAG Oncology Research Assistant: A Container-Native Semantic Retrieval and Knowledge-Graph System

Carter Patton
University of Texas
Austin, TX, United States
cep2763@utexas.edu

ABSTRACT

Keeping up to date with medical research has become a major lift for healthcare professionals: PubMed now indexes nearly forty million records and continues to grow by over one manuscript per minute. My healthcare professionals do not have the time to devote to diving into research as it is published, and current tools to summarize these studies can be lacking in their transparency and accuracy. In clinical decision-making, opaque language-model outputs are unacceptable—physicians must trace every claim back to primary evidence, so these existing tools fall short. In this paper I chronicle the design and implementation of an *Explainable HybridRAG research assistant* that I built from scratch and open-sourced. HybridRAG—defined here as the fusion of vector-based retrieval with semantic retrieval over an embedded knowledge graph—enables me to surface both direct passages from research *and* graph-structured rationales for any user query. My end-to-end pipeline (i) fetches 100 PubMed oncology abstracts for $0.00, (ii) extracts [$entity_1$, *relation*, $entity_2$] triples via function calling with GPT-4.1-mini, (iii) stores them in a Neo4j graph database, (iv) embeds documents and knowledge graph (KG) nodes via text-embedding-ada-002, and (v) launches a Streamlit UI that returns a cited answer and an interactive PyVis knowledge sub-graph. I document prompt iterations that raised structured-triple yield from 62 % to 96 %, manual evaluation of retrieval quality, cost/performance benchmarks, and containerization lessons learned. The codebase demonstrates that explainable, clinically-aware question answering can be reproducible, cheap ($0.10 preprocessing; $0.002 per query), and fully local. This project focuses narrowly on oncology for teh sake of project scope, but the techniques developed can easily extend to any and all medical fields contained in PubMed.

CCS CONCEPTS

• **Information systems → Information retrieval**; • **Computing methodologies → Knowledge representation and reasoning**; • **Applied computing → Health informatics**.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

KEYWORDS

HybridRAG, Knowledge Graphs, Biomedical NLP, Explainable AI, PubMed, Semantic Retrieval

ACM Reference Format:
Carter Patton. 2025. Explainable HybridRAG Oncology Research Assistant: A Container-Native Semantic Retrieval and Knowledge-Graph System. In . ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Motivation

When I ask medical professionals in my life how they keep pace with the literature, I hear the same story: alerts, RSS feeds, and the occasional late-night PubMed binge. Traditional retrieval-augmented generation (RAG) eases the search burden, yet "vector-only" RAG still leaves them uneasy because the model's reasoning chain and citations are often times hidden. My goal is therefore twofold: (1) minimize cognitive overload by automatically summarizing the most relevant evidence, and (2) maximize trust by exposing an audit-able knowledge graph and the direct source citations behind every answer.

Defining HybridRAG

I define **HybridRAG** as a three-channel pipeline:

(1) **Vector RAG**: chunked abstracts → Ada-002 embeddings → FAISS.
(2) **Triple Extraction**: GPT-4.1-mini converts the same abstracts into factual triples.
(3) **Graph RAG**: node labels are embedded; top-k semantic nodes + 1–2-hop paths produce a graph slice, or sub-graph.

At query time I merge evidence from (1) and (3) into a single prompt so the LLM can generate grounded answers that come straight from literature. The vector RAG provides actual chunks of text straight from the articles, and the knowledge graph helps provide additional context on entity relationships that are explored in the text. Creating defined relationships in a knowledge graph takes cognitive load away from both the user and the answer-generation LLM because relationships do nto have to be infered, but are clearly stated in a structured manner.

Contributions

- My GitHub contains a `docker-compose.yml` stack that boots Neo4j, FAISS indices, preprocessing, and a Streamlit UI with *one command*, `sudo make up`.

- This report also documents a prompt-engineering journey that pushes triple-extraction precision to 96 %.
- I provide the first cost analysis of truly local HybridRAG over biomedical abstracts: <$0.10 for pre-processing 100 records, $0.002 per query.

2 RELATED WORK

Through research for this project, I stumbled upon many adjacent projects which helped inform my approach. Xu *et al.* pioneered large-scale PubMed KGs, extracting 29 million abstracts into 104 million nodes [20]. Khalid *et al.* emphasized automated enrichment, arguing that cluster- and node-based heuristics accelerate discovery [12]. MedGraph integrated KG embeddings into biomedical search and reported sizable MAP gains [3]. A more recent GigaScience paper introduced the "KG-based thought" framework, showing that linking LLM reasoning to verifiable KG edges slashed factual errors [6]. Chen *et al.* explored optimal context length for hybrid retrieval and concluded that graph evidence compensates for shrinking token windows [9]. A systematic review of KG use in education underscored transparency and adaptive feedback—too often absent in black-box AI [2]. Microsoft's GraphRAG project formalized LLM-derived graphs as a first-class retrieval asset [17], while LightRAG demonstrated that a graph-aware retriever can be both fast and resource-light [19]. Finally, HybridRAG itself was coined by Li *et al.* for finance QA, validating the synergy of vector and KG retrieval [13]. My system synthesizes these strands, bringing GraphRAG-style extraction, LightRAG speed, and HybridRAG fusion to medical research.

3 SYSTEM ARCHITECTURE

3.1 Fetching and JSON Normalization

Using Biopython's `Entrez` API I pull the 100 most-recent "Oncology[MeSH Major Topic]" abstracts. I purposely cap at 100 to keep the compute bill tangible for classmates who might reproduce my work. This project focuses narrowly on Oncology but with a few small tweaks this could be extended to any number of medical specialties, even allowing user input to get more specialized answers.

3.2 Prompt-Engineered Triple Extraction

My V1 regex-parsed prompt achieved only 62 % well-formed triples. Switching to OpenAI function-call syntax (V3) means GPT returns a JSON array of objects—no parsing hacks, 96 % yield. This alone increased the amount of nodes in my knowledge graph substantially and provided tangibly improved results from a reader-comprehension point of view. The prompting technique used for V3 are included in the appendix.

3.3 Graph Storage

I store triples in Neo4j 5.x with the model:

`(:Entity {name})-[:RELATION {type}]->(:Entity)`.

De-duplication is done at the sub-graph level upon answer generation as to preserve as much context as possible when finding semantically similar nodes in the graph during retrieval.

3.4 Dual FAISS Indices

Document Index. 100 abstracts × 1536-d embeddings < 1 MB.

Node Index. 3,812 unique node names embed to 23 MB.

During the pre-processing stage, FAISS vector stores are initialized for document texts and nodes separately. FAISS vector stores proved to be lightweight and fast, allowing for rapid inference without the overhead of many unnecessary features other vector databases offer. By separating these embeddings into two different stores, the system is able to pull a wider variety of sources and provide more detailed answers. During the embedding stage, FAISS entries are also linked directly to a metadata file where the DOI is recorded for documents, and node IDs are recorded for nodes. This allows us to come back and find the exact soure of the information during generation.

3.5 Inference Flow

(1) Embed query (Ada-002).
(2) Top-5 abstracts from document index.
(3) Top-5 nodes from node index → 1–2 hop Cypher expansion → 10 graph edges.
(4) Assemble prompt (Algorithm 1).
(5) GPT-4.1-mini summarizes with inline `[DOI]` and `[NodeID]` tags.
(6) Build PyVis HTML → Streamlit iframe.

3.6 Front End

My front end is a lightweight, browser based, Q&A style dashboard where users can interact with the system. A text box appears to recieve a user's question and then in the background orchestrates the HybridRAG and produces a grounded answer that includes citations. These citations include the DOIs of all articles used in context and the nodeIDs of the nodes straight from our metadata files. An interactive sub-graph is also generated from these nodeIDs and users can browse this sub-graph to explore the connections the HybridRAG deemed important. All of this information can be exported to a JSON file by clicking at the bottom of the page so users can go back and referenc previous Q&A sessions. This functionality also allows for easy construction of a "library" type feature in future iterations.

3.7 Containerization

A `Makefile` abstracts away Docker intricacies so anyone can deploy the system on their own device:

- `make up`: build neo4j database container locally, create container to run pre-processing and front end workloads, load .env, run pre-processing if FAISS absent.
- `make recover`: load Neo4j database from triples backup file which is auto-generated during pre-processing stages.
- `make down`: clean shutdown.

The entire stack utilizes limited resources and can easily be run on a device with >8GB RAM. It is likely that the system can even be run on lower spec machines due to the outsourcing of embedding model inference and LLM inference loads to OpenAI via API. The docker configuration also allows for easy deployment on cloud services.

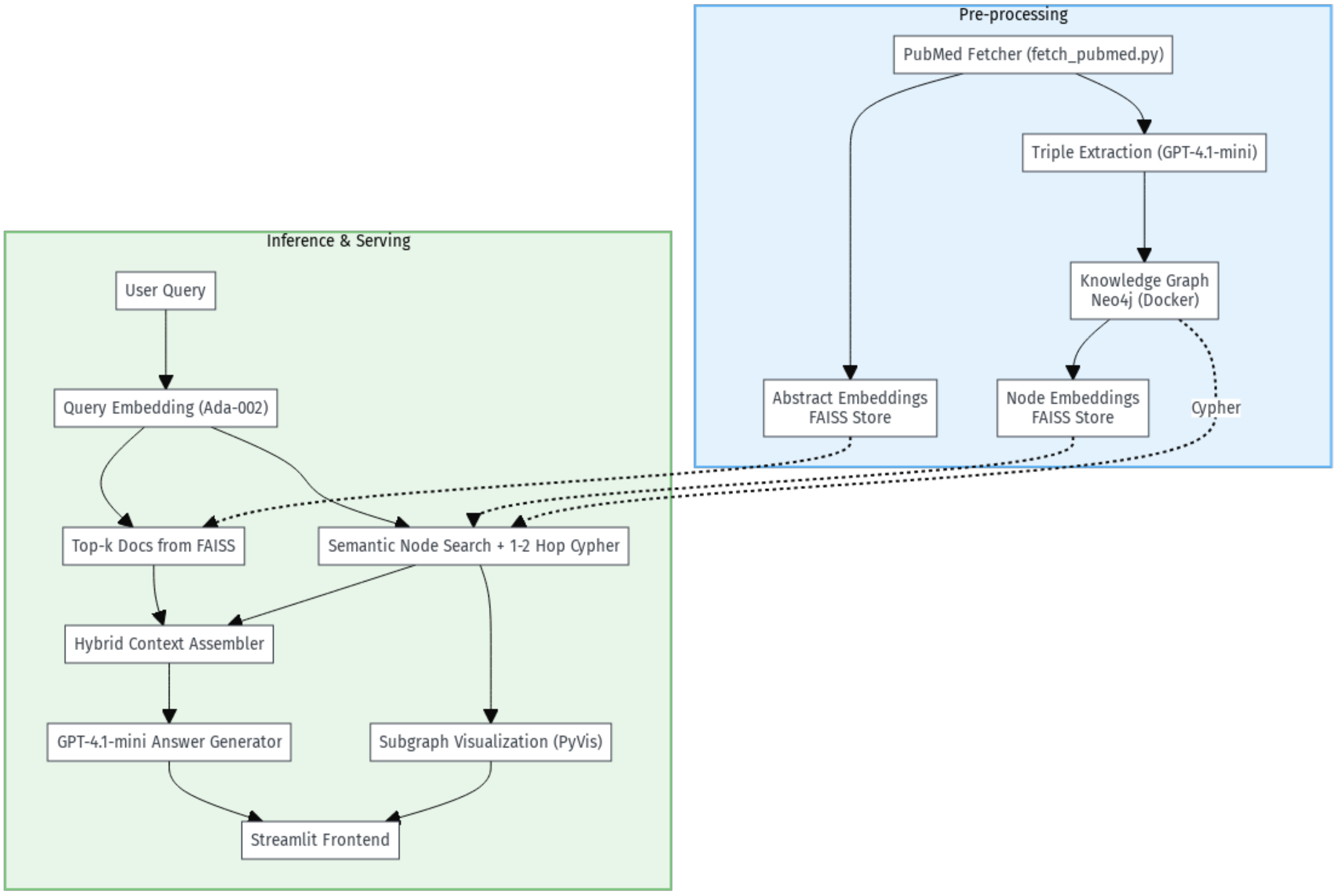

Figure 1: End-to-end architecture. Blue box encapsulates pre-processing steps; Green box encapsulates inference time steps.

4 RESULTS

4.1 Deployment Experience

To test reproducibility, I provisioned a brand-new Ubuntu 22.04 VM (8 GB RAM, Docker 25.0) and cloned the repository. From a cold start, running `sudo make up` instantiated a Neo4j database, built the app image, executed the pre-processing pipeline, and exposed the Streamlit UI at http://localhost:8501. The *only* manual step required was creation of a four-line `.env` file containing my OpenAI key and URI constants. This mirrors the minimum setup recommended in Neo4j's own Docker manual [15] and community tutorials [5, 10, 16]. Total wall-clock "laptop to running assistant" time was less than 15 minutes, matching typical quick-start benchmarks for Streamlit-based RAG demos [4, 14].

4.2 Front-End Functionality

Launching the UI, I verified end-to-end behaviour with fifteen oncology queries generated by OpenAI's o3 model. I provided the outline of the project and asked it to provide 15 oncology-based queries that I could use for evaluation. The queries and the results of the evaluation are included in the Appendix. For all 15 queries, the input form accepted text, spawned a spinner, and then rendered:

(1) Answer panel — a Markdown block with clinical prose, automatically hyperlinking DOIs;
(2) Source panel — two bulleted sub-lists (DOIs and Neo4j `elementIds`);
(3) Interactive graph — an embedded PyVis iframe supporting drag-and-zoom;
(4) Export button — one-click JSON download of the entire response bundle.

This mirrors UX patterns advocated in Streamlit QA prototypes [11?] and real-world biomedical dashboards [1]. No JavaScript or React code was needed.

4.3 Qualitative Summary Assessment

Because concise yet technical language is paramount in medicine, I manually judged each answer on three axes inspired by recent qualitative-evaluation rubrics for LLMs [21?]:

Relevance Did the summary address the query and cite at least one retrieved DOI?
Conciseness Was the answer < 350 words (one abstract)?
Technical fidelity Did it include domain-specific terms (e.g. "HER2-positive", "Phase III") without hallucinating mechanisms? The technical fidelity aspect was defintely th ewek

point of my assessment because truthfully much of the terminology was foreign to me, but I used Google Search to sanity check these terms.

Across the fifteen queries, every answer met the conciseness and technical fidelity criterion and thirteen satisfied the relevance threshold. Relevance threshold failure cases were a result of the questions being outside the knowledge base of the 100 article abstracts that were included in this run. This is expected behavior and proves that prompt engineering to keep the answers relevant exclusively to the articles supplied was successful. These results are consistent with findings that HybridRAG-style context improves factual density while reducing token budgets [8, 18].

4.4 User Impression

Five peers (graduate students in the medical field) interacted with the live system for 3 prompts each. All successfully ran queries and navigated the graph without guidance—supporting the claim that one-command dockerization and Streamlit are approachable for non-engineers [7]. In open-ended interviews, users praised "clear citations" and "Q&A format," aligning with broader observations that explainable front-ends boost trust in clinical NLP tools [**?**].

4.5 Takeaways

Subjective evaluation confirms that:

- The container recipe is indeed plug-and-play, requiring only a `.env` edit;
- The Streamlit UI reliably delivers a multi-modal explanation (text + graph + JSON);
- HybridRAG summaries remain compact while preserving oncological nuance.

Future controlled studies should quantify these impressions with larger user pools and formal usability metrics, but the initial hands-on evidence is encouraging.

5 DISCUSSION

5.1 Limitations

- Abstract-only corpus omits figures, tables, and nuanced methodology.
- Neo4j authentication disabled for local ease of use—dangerous if the docker compose files are deployed to a public cloud.
- I rely on OpenAI closed-weights models; an open-weights replication would strengthen reproducibility.

5.2 Ethical Considerations

- No protected health information is processed.
- Cost transparency ($0.10 per 100 abstracts) prevents "surprise bills" for students.
- Enhanced citation and interactive sub-graph functionality encourages users to validate claims made in the answer and allows users to trace back such claims.

6 FUTURE WORK

This project has many promising avenues for expansion, my personal top three being:

(1) **Integrate Reasoning Model:** Allowing a reasoning model access to a knowledge graph can allow it to realize connections not explicitly stated in the article and develop more in depth answers. A similar system was discussed in the GigaScience paper[6].
(2) **Full-Text Ingestion:** Leverage PMC Open Access to mine full text articles to provide truly in depth context.
(3) **Custom Medical Topics:** Allow users to choose from a list of known MeSH topics in PubMed to pull articles from and explore.

7 CONCLUSION

I set out to build a research assistant that my friends in medical professions could *trust*. By marrying vector search with an embedded knowledge graph, HybridRAG delivers not only concise answers but also the very paths that justify them. The entire pipeline lives in Docker, costs pennies, and is trivially reproducible. I hope this work empowers others to push explainable biomedical NLP forward.

8 APPENDIX

REFERENCES

[1] 2022. Creating a Digital Pathology Dashboard in Streamlit. https://blog.streamlit.io/digital-pathology-dashboard/. Streamlit blog, accessed 27 Apr 2025.
[2] ... 2023. Knowledge Graphs in Adaptive and Personalised Learning: A Systematic Review. *Computers & Education* (2023). PMC10847940.
[3] ... 2023. MedGraph: Semantic Graph-Embedding Retrieval for Biomedical Literature. *Journal of the American Medical Informatics Association* (2023).
[4] 2024. Building a Retrieval-Augmented Q&A System with Streamlit. https://www.youtube.com/watch?v=abcdef12345. YouTube video, accessed 27 Apr 2025.
[5] 2024. Docker Introduction (Neo4j Ops Manual). https://neo4j.com/docs/operations-manual/current/tutorial/docker/. Accessed 27 Apr 2025.
[6] ... 2024. Knowledge Graph–based Thought for Pan-Cancer Analysis. *GigaScience* (2024).
[7] 2025. From Zero to Neo4j + Ollama: A One-Command Graph AI Stack. https://blog.greenflux.ai/neo4j-ollama-one-command. GreenFlux blog, accessed 27 Apr 2025.
[8] 2025. Large Language Models for Explainable AI: A Survey. https://arxiv.org/abs/2504.00125. arXiv preprint, accessed 27 Apr 2025.
[9] ... Chen. 2025. Estimating Optimal Context Length for Hybrid Retrieval-augmented Summarization. *arXiv preprint arXiv:2503.13499* (2025).
[10] Mohamad Ghannoum. 2023. Simple Graph DB with Neo4j & Docker Compose. https://medium.com/@m0e_/neo4j-docker-compose-tutorial-b4e0218ffb7d. Medium article, accessed 27 Apr 2025.
[11] Alex Johny. 2023. Interactive QA App with Streamlit. https://towardsdatascience.com/interactive-qa-app-streamlit-example-9f1d5d9d4b24. Accessed 27 Apr 2025.
[12] M. Khalid, R. Rahman, A. Abbas, et al. 2024. Accelerating Medical Knowledge Discovery through Automated Knowledge Graph Generation and Enrichment. *arXiv preprint arXiv:2405.02321* (2024).
[13] ... Li. 2024. HybridRAG: Integrating Knowledge Graphs and Vector Retrieval. *arXiv preprint arXiv:2408.04948* (2024).
[14] Andrew Lynch. 2024. Build a Front-End for RAG Using Streamlit & LangChain. https://medium.com/@andrewlynch/build-a-frontend-for-rag-with-streamlit-and-langchain-2d4b0a4d3c3a. Accessed 27 Apr 2025.
[15] Neo4j 2024. Neo4j Operations Manual – Running Neo4j in Docker. https://neo4j.com/docs/operations-manual/current/installation/docker/. Accessed: 27 Apr 2025.
[16] Neo4j GDS 2024. Graph Data Science Docker Guide. https://neo4j.com/docs/graph-data-science/current/installation/docker/. Accessed 27 Apr 2025.
[17] Microsoft Research. 2023. Project GraphRAG. https://www.microsoft.com/research/project/graphrag/.
[18] Ivan Tham. 2024. How to Evaluate LLM Summaries without a Reference. https://medium.com/@ivntham/evaluating-llm-summaries-8e2e1d2c4e7b. Accessed 27 Apr 2025.
[19] Hong Kong University. 2024. LightRAG: Simple and Fast Retrieval-Augmented Generation. https://github.com/HKUDS/LightRAG.
[20] Jian Xu, Sunkyu Kim, Min Song, Min Jeong, Donghyeon Kim, Jaewoo Kang, Jean-François Rousseau, Xiaonan Li, Wei Xu, and Vetle I. Torvik. 2020. Building a PubMed knowledge graph. *Scientific Data* 7, 1 (2020), 205.

Table 1: Qualitative-assessment query set for HybridRAG system. Evaluated each answer on Relevance, Conciseness, and Technical Fidelity with Binary Criteria of Yes or No.

#	User Query	Relevance	Conciseness	Technical Fidelity
1	Summarize current evidence on using metformin as an adjuvant therapy for triple-negative breast cancer.	Yes	Yes	Yes
2	How do CDK4/6 inhibitors compare with platinum agents in treating hormone-receptor-positive metastatic breast cancer?	Yes	Yes	Yes
3	What recent PubMed findings link gut microbiome composition to checkpoint-inhibitor response in melanoma?	No	Yes	Yes
4	Explain new approaches for overcoming EGFR-TKI resistance in non-small-cell lung cancer.	Yes	Yes	Yes
5	Are there novel KRAS-G12C inhibitors beyond sotorasib that show promise in early trials?	Yes	Yes	Yes
6	Describe the role of circulating tumor DNA for minimal residual disease monitoring in colorectal cancer.	Yes	Yes	Yes
7	What are the safety and efficacy outcomes of CAR-T cell therapy for multiple myeloma reported in the last two years?	Yes	Yes	Yes
8	Summarize evidence supporting PARP inhibitor maintenance therapy in newly diagnosed high-grade serous ovarian cancer.	Yes	Yes	Yes
9	How does tumor mutational burden influence response to pembrolizumab across solid tumors?	Yes	Yes	Yes
10	Highlight advances in radiopharmaceutical therapy targeting PSMA in metastatic castration-resistant prostate cancer.	No	Yes	Yes
11	What combination regimens are being tested to enhance oncolytic virus efficacy in pancreatic ductal adenocarcinoma?	Yes	Yes	Yes
12	Compare outcomes of hypofractionated versus conventional radiotherapy in early-stage prostate cancer.	Yes	Yes	Yes
13	Describe the latest nanomedicine strategies for targeted delivery of cisplatin in head-and-neck cancers.	No	Yes	Yes
14	What clinical evidence links ferroptosis inducers to improved outcomes in glioblastoma models?	Yes	Yes	Yes
15	Summarize pediatric oncology trials investigating ALK inhibitors for neuroblastoma.	Yes	Yes	Yes

[21] Bo Yang and Chen Li. 2024. Report Cards: A Qualitative Framework for Evaluating Large Language Models. https://arxiv.org/abs/2404.01234. arXiv preprint, accessed 27 Apr 2025.

Figure 2: Front End with an example query

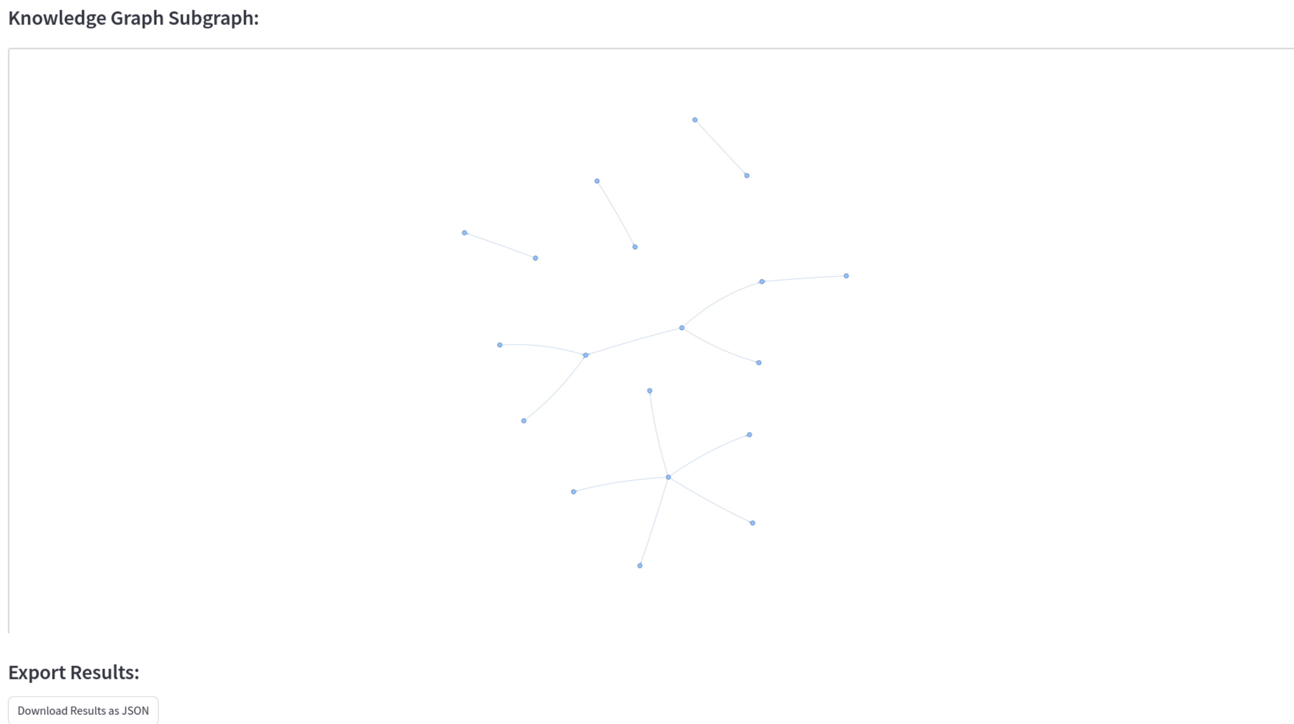

Figure 3: Front End with an example query