

The Anatomy of a Trustworthy AI Answer: A Comparative Experiment for RAG Architectures

Dippu Kumar Singh¹, Praveen Chinapla Bharamappa²

¹Fujitsu North America Inc.

²Applied Materials

dippu.singh@fujitsu.com, praveen.chinapla@gmail.com

Abstract

Retrieval-Augmented Generation (RAG) has become the go-to fix for LLM hallucinations. But its most common form, built on Vector Databases, is like a confident consultant who has only read the executive summaries. It's fluent, convincing, and adept at finding information that sounds right, but critically lacks the deep, verifiable connections between the facts. In high-stakes domains like medicine, this creates a dangerous new form of AI: one that is wrong with conviction. This paper provides a comparative experiment for distinguishing between answers that merely sound correct and those that are verifiably true. Our head-to-head evaluation of Vector-based versus Knowledge Graph-based RAG reveals a stark architectural choice. Our findings demonstrate that while Vector RAG produces a convincing but untraceable story, the Knowledge Graph approach delivers a factually correct answer with a verifiable evidence trail. This is the blueprint for building RAG systems that don't ask for your trust - they earn it by showing their work.

Introduction

The enterprise world is in a frenzied race to adopt Generative AI. According to Gartner, "By 2026, more than 80% of enterprises will have used generative artificial intelligence (GenAI) application programming interfaces (APIs) or models up from less than 5% in 2023" (Gartner, 2023). But this stampede towards innovation is heading directly into a minefield. In high-stakes sectors like healthcare, finance, and legal services, the adoption of these powerful models is being crippled by their three cardinal sins: inaccuracy, hallucination, and a critical lack of explainability. The prevailing response has been to ground these models using vector-based Retrieval-Augmented Generation (RAG). However, this is a blunt instrument. While effective at retrieving factual fragments, vector search is fundamentally incapable of understanding the complex, multi-hop relationships that define enterprise knowledge. It can find a needle in a haystack,

but it cannot explain how the needle is connected to the thread. This paper argues for a more intelligent foundation. We posit that Knowledge Graphs (KGs) are the essential, structured backbone required to move from simple retrieval to complex, verifiable reasoning (Pan, 2023). KGs provide a "neural map" of an organization's domain, allowing GenAI not just to find information, but to reason over it. This work moves beyond theory to present a validated experiment that demonstrates how a KG-based approach delivers a superior capability for complex reasoning when compared to traditional vector-based search. It provides an architectural blueprint for solution architects, data scientists, and business leaders seeking to build the next generation of trustworthy, explainable, and verifiably intelligent enterprise AI.

Existing Key Challenges

The global large language model (LLM) market was valued at \$4.35 billion in 2023 and is projected to grow at a compound annual growth rate (CAGR) of 35.9% through 2030 (Grand View Research, 2024). However, this rapid expansion relies on a precarious technological foundation. As detailed in Figure 1, several underlying issues exist, but four key challenges in particular currently prevent the safe and reliable deployment of LLMs in high stake environments.



Figure 1: Key Challenges in adopting GenAI (k2view, 2025)

- **A Crisis of Veracity (Hallucinations):** LLMs are prone to generating factually incorrect, biased, or nonsensical content, known as "hallucinations." This presents a critical and unacceptable risk in any fact-based enterprise, from legal analysis to medical diagnostics.
- **A Crisis of Context (Domain Blindness):** These models are a mile wide and an inch deep. They lack the deep, nuanced understanding of specialized domains, rendering their responses superficial and unreliable for expert use cases where precision is paramount.
- **A Crisis of Timeliness (Stale Data):** LLMs are temporal snapshots, frozen in time at the moment their training data was collected. They are incapable of accessing real-time information that is the core of dynamic business environments, making them inherently outdated.
- **A Crisis of Transparency (The Black Box):** They operate as "black boxes," unable to explain their reasoning or cite their sources. This lack of explainability makes them unusable in any regulated industry where accountability and auditability are non-negotiable.

These are not mere technical hurdles; they are the four horsemen of enterprise AI, and they represent a fundamental barrier to trust that must be overcome.

Solution Design

Solution Overview

This paper evaluates a hybrid Retrieval-Augmented Generation (RAG) architecture that compares a Knowledge Graph with a Vector Database. We specifically measure the improvement in response accuracy provided by the Knowledge Graph component.

Vector Database and Knowledge Graphs Overview

To ground a Large Language Model in reality, one must first give it memory. In the field of enterprise AI, two fundamentally different philosophies have emerged for how to construct this memory. The first, and most common, is the Vector Database, which can be understood as a semantic scrapbook. This approach effectively shreds unstructured documents into fragments, converts each fragment into a mathematical vector embedding, and stores them in a high-dimensional index (Pinecone Systems Inc., 2023). Its power lies in semantic similarity search: it can instantly find all the page fragments that are conceptually similar to a user's query.

However, its greatest strength is also its fatal flaw: in the process of indexing, the vector database obliterates the structural context and the explicit relationships between the data. It knows what is on the pages, but it has no memory of how they connect. The second philosophy is the Knowledge Graph (KG), which is not a scrapbook but a neural map of the enterprise. Instead of storing disconnected fragments, a

KG captures a structured representation of real-world entities (people, products, documents) and, crucially, the explicit relationships that connect them (Hogan, 2021). This creates a navigable network of information. This structure empowers the KG to handle multi-hop queries that require a deep, contextual understanding of the data, allowing the AI to move from simple retrieval to genuine reasoning.

System Requirements

The modules required for this experiment are as follows:

Method	Modules
Vector-based Method	Python, LangChain, tiktoken, recursive text splitter, ChromaDB
Knowledge Graph-based Method	Python, LangChain, tiktoken, recursive text splitter, Neo4j, OpenAI, spacy

Table 1: Modules for the solution

Prerequisites required for this experiment are Neo4j account (Neo4j Account Creation) and OpenAI account (OpenAI Account). We have used a free Neo4j instance for the experiment, which is hosted on the cloud, but this can be also hosted locally to keep the KGs in local environment (Neo4j Deep Understanding).

Data Preparation

Prior to scraping, all source URLs were manually reviewed to ensure they originated from medically accredited authors and publications. The raw HTML content was cleaned to remove navigational elements, and advertisements to isolate the core article text for the pre-processing pipeline.

Data Sourcing

To create a fair and demanding benchmark, we deliberately selected a dataset from one of the most unforgiving domains for AI: healthcare. In this high-stakes environment, factual correctness is not a feature; it is a non-negotiable prerequisite, and the penalty for hallucination is severe. For our experiment, we sourced and scraped a comprehensive corpus of data on "Diabetes" from authoritative public medical source (Healthline, 2025).

This domain was purposefully chosen for our experiment. It allowed us to test each system's ability to move beyond simple semantic retrieval to the complex, relational reasoning required for medical knowledge, where understanding the connections between symptoms, treatments, and contraindications is paramount. The public data was acquired using the open-source web scraping tools Selenium and Beautiful Soup.

Data Pre-processing

To ensure a fair and reproducible comparison, both the vector-based and graph-based architectures were fed by a single, standardized data processing pipeline. This pipeline, illustrated in Figure 2, acts as a "digital scribe," systematically transcribing the raw, unstructured text from our public data source (Healthline, 2025) into a uniform, tokenized format ready for ingestion.

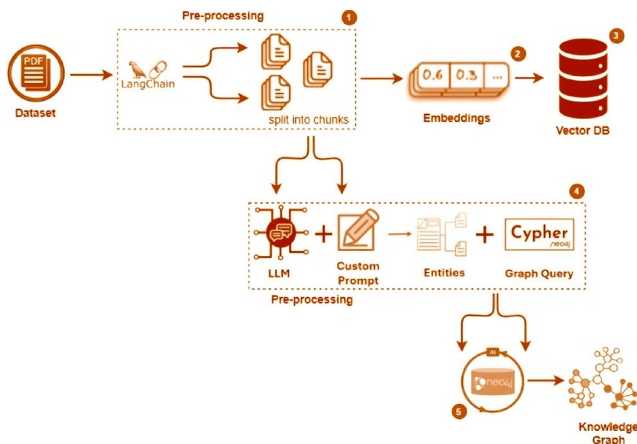


Figure 2: Data Pre-processing and Storing in Knowledge Base

The core of this pipeline is a chunk generator, implemented using the LangChain framework (Ketonen, & Tsilonis, 2023). This function is tasked with a critical preliminary step in any RAG system: breaking down large, unwieldy documents into smaller, semantically coherent chunks. As detailed in Algorithm 1, we configured the chunk generator with a `chunk_size` of 4096 characters (64x64) and a `chunk_overlap` of 40 characters. This overlap is an engineering choice, designed to preserve the contextual continuity between consecutive chunks, preventing a logical idea from being severed during the splitting process (Brown, 2020).

Algorithm 1: Chunk Generator

```
python_splitter = RecursiveCharacterTextSplitter.from_language(
    language=Language.PYTHON,
    chunk_size=64*64,
    chunk_overlap=40,
)
```

Following the chunking process, each text fragment is passed to the tiktoken library to calculate the precise number of tokens it contains. This tokenization step is the final common process. From this point, the standardized, chunked data is forked, with one stream being sent to the vector database for embedding and the other to the knowledge graph for triple extraction, as detailed in the following sections.

Vector database and Knowledge Graph Creation

With the data processed into a standardized format, we now arrive at the central divergence of our experiment: the construction of the AI's long-term memory. This section details the two competing blueprints for transforming raw text chunks into a query-based knowledge base.

The Vector DB Instance:

The first approach, illustrated in Steps 2 and 3 of Figure 2, is a canonical implementation of a vector-based RAG system. Each standardized text chunk is passed through an embedding model, which transforms the unstructured text into a high-dimensional numerical vector. This embedding captures the semantic essence of the text. These vectors are then ingested and indexed into our vector database, ChromaDB. The result is a "semantic library" - a searchable space where the proximity between vectors corresponds to the conceptual similarity of the text they represent (ChromaDB, 2023).

The Knowledge Graph Instance:

The second approach, illustrated in Steps 4 and 5 of Figure 2, is a more complex but also more powerful process. Instead of creating a simple semantic index, this pipeline is designed to construct a structured, relational Knowledge Graph. The core of this process is the LLMGraphTransformer, a module within the LangChain framework that uses a powerful LLM - in our case, GPT-4o to act as a knowledge architect (OpenAI, 2023). The transformer systematically analyzes each text chunk, identifies the core entities (e.g., "Insulin," "Type 2 Diabetes"), and, crucially, extracts the explicit relationships that connect them (e.g., "is a treatment for"). This structured output of entities and their relationships is then translated into a Cypher query and ingested into our Neo4j graph database. The result, as shown in Figure 3, is not a library of disconnected fragments, but a true neural map of the domain - a network of interconnected nodes and relationships that represents a far deeper and more structured form of knowledge.

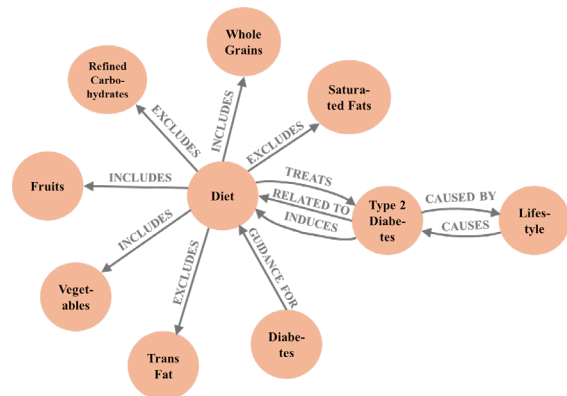


Figure 3: Visualizing the Type 2 Diabetes node and its connections in a Knowledge Graph.

RAG Data Retrieval Method

Retrieval-Augmented Generation (RAG) is the foundational technique for grounding Large Language Models in reality, providing them with an external knowledge base to make their responses more accurate, timely, and less prone to hallucination. However, the method by which the RAG system retrieves this knowledge is a critical architectural choice that fundamentally defines the system's intelligence.

The vector database employs a strategy of semantic search. When a user poses a query, that query is first transformed into an embedding vector, Q . The database then performs a high-speed similarity search against the stored document chunk vectors, D , to find the chunks with the highest conceptual similarity (Gao, 2023). This is typically calculated using cosine similarity:

$$\text{Similarity}(Q, D) = \frac{Q \cdot D}{|Q||D|} = \frac{\sum_{i=1}^n Q_i D_i}{\sqrt{\sum_{i=1}^n Q_i^2} \sqrt{\sum_{i=1}^n D_i^2}}$$

Where:

- Q is the query vector.
- D is the document chunk vector.
- The result is a score between -1 (opposite meaning) and 1 (identical meaning).

The top k document chunks with the highest similarity scores are then retrieved and passed to the LLM as context. This is the informational equivalent of a librarian handing you a stack of relevant but unordered pages.

The Knowledge Graph, in contrast, enables relational reasoning. Because the data is stored as a structured network of nodes and relationships, the retrieval process is a far more sophisticated, multi-modal query. It can leverage:

- **Keyword and Vector Search:** To identify the most relevant starting nodes or relationships in the graph.
- **Graph Traversal:** To navigate the network starting from those initial nodes, discovering additional, contextually relevant information that would be invisible to a simple similarity search (Johnson, Douze, & Jégou, 2019).

For instance, let's say our Knowledge Graph G is a set of nodes and relationships. We want to find the "treatment" for "Diabetes." The query pattern, which is the KG's equivalent of a formula, would be:

$$(d:\text{Disease}) \rightarrow (t:\text{Treatment})$$

This pattern instructs the database to:

- Find a node d with the label Disease and the property name equal to 'Diabetes'.
- Follow an outgoing relationship.
- Return the node t at the end of that relationship, which has the label Treatment.

This is like the informational equivalent of a librarian handing you the right book and showing you the entire section of the library to which it belongs. By providing the

LLM not just with isolated facts, but with a connected KG-based RAG empowers a deeper, more reliable, and more explainable form of reasoning (Angles, & Gutierrez, 2008).

Vector Database Retrieval Process

The vector-based architecture operates on a two-stage principle: first, a high-speed semantic search to find relevant information, followed by an LLM-powered synthesis to construct a coherent answer. The end-to-end process is illustrated in Figure 4.

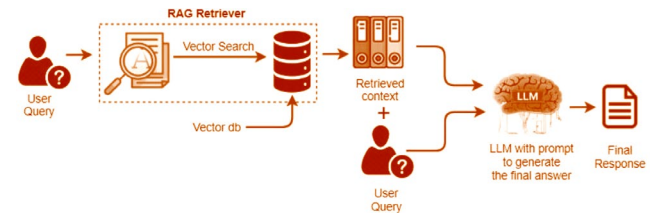


Figure 4: The RAG retrieval and generation process for the vector-based method.

The process begins when a user submits a query. The RAG system first converts this natural language question into a high-dimensional embedding vector. This query vector then becomes the key to unlocking the knowledge stored in the ChromaDB vector database.

The system executes a similarity search, retrieving the top- k most relevant document chunks based on the cosine similarity between the query vector and the stored vectors (Johnson, Douze, & Jégou, 2019). This leverages ChromaDB's powerful indexing capabilities to find the fragments

Algorithm 2: ChromaDB Query for top- k similar vectors

```
query_results = collection.query(
    query_texts=["What is the relationship between Diabetic Ketoacidosis and Kelly Wood, Md?"],
    n_results=1)
```

Algorithm 3: The function call to generate a grounded response using the retrieved context.

```
"""user query and prompt pass to LLM"""
1: def get_endpoint_connection(prompt):"""
chatAI model connection"""
client = AzureOpenAI(azure_endpoint="<<_url_>>",
api_version="2023-05-15",
api_key=API_KEY,
http_client=httpx.Client(event_hooks={"request":
[chat_completion]}),)
2: conn = client.chat.completions.create(messages=
[{"role": "user", "content": "{prompt}"}], model="gpt-4o",)
3: return conn.choices[0].message.content
```

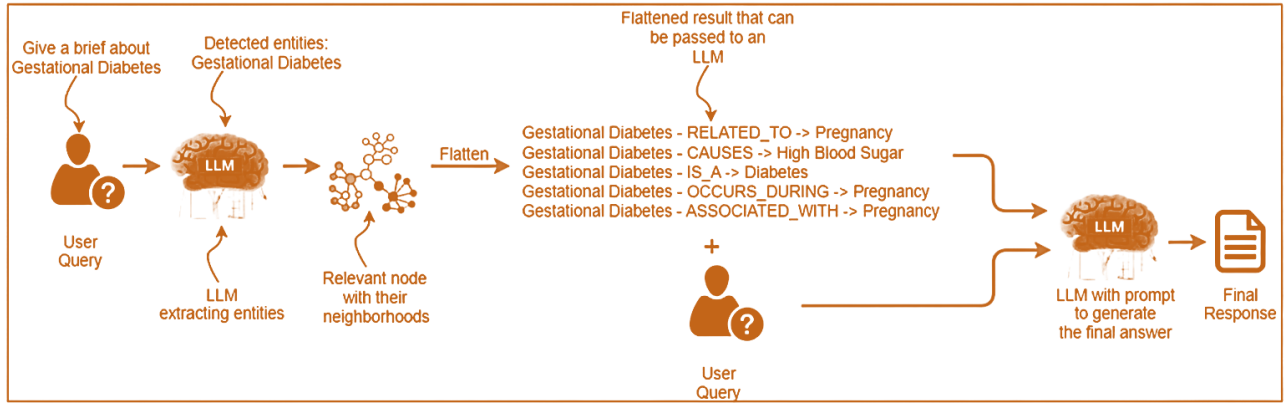


Figure 5: The RAG retrieval and generation process for the Knowledge Graph method.

of text that are closest in semantic meaning to the user's intent, as shown in the query implementation in Algorithm 2.

In the second stage, these retrieved but unordered chunks of text are passed as context to the final LLM, GPT-4o. The model is then given a single, critical instruction: synthesize a comprehensive answer to the original user query, relying only on the provided context. This crucial step of grounding the LLM prevents it from relying on its internal, pre-trained knowledge, thereby minimizing hallucinations and ensuring the final response is directly traceable to the source documents (OpenAI, 2023). The implementation of this final synthesis step is shown in Algorithm 3.

Knowledge Graph Retrieval Process

In contrast to the single-step similarity search of a vector database, the Knowledge Graph architecture operates as a multi-stage reasoning engine. It mimics a human expert's workflow: first, deconstructing the question; querying a structured knowledge base; and synthesizing a comprehensive answer. The entire process is illustrated in Figure 5.

1. Query Decomposition and Entity Extraction: The process begins not with a search, but with an act of understanding. When a user submits a question, it is passed to the LLM

Algorithm 4: The prompt template for LLM-based entity extraction

```
prompt = ChatPromptTemplate.from_messages(
    ["system",
     "You are extracting organization and person entities
    from the text.",
     ("human",
      "Use the given format to extract information from
    the following "
      "input: {question}"),
    ]
)
```

tasked with Named Entity Recognition (NER). Using a crafted prompt template, as shown in Algorithm 4, the LLM deconstructs the natural language query and extracts the core entities to form the basis of our graph query (Li, 2022).

2. Structured Knowledge Retrieval: The extracted entities are systematically injected into a Cypher query template, as shown in Algorithm 5. This transforms the unstructured user question into a formal, structured query that can be executed against our Neo4j graph database (Neo4j, 2023).

Algorithm 5: The Cypher query template for graph traversal.

```
# User's question is passed to extract entities and then
nodes from KG based on entities
1: def entity_based_context_retriever(question: str) -> str:
2: result = ""
3: entities = entity_chain.invoke({"question": question})
4: for entity in entities.names:
    response = graph.query("""CALL db.index.fulltext.queryNodes('entity', $query, {limit:2})
    YIELD node, score
    CALL {
      WITH node
      MATCH (node)-[r:!MENTIONS]->(neighbor)
      RETURN node.id + '-' + type(r) + '->' + neighbor.id AS output
    UNION ALL
    WITH node
    MATCH (node)<-[r:!MENTIONS]-(neighbor)
    RETURN neighbor.id + '-' + type(r) + '->' + node.id AS output
    }
    RETURN output LIMIT 50""",
    {"query": entity},
    )
5: result += "\n".join([item['output'] for item in response])
6: return result
```

3. Contextual Graph Traversal:

Unlike a vector search that returns isolated chunks, the Cypher query returns a rich, interconnected subgraph of relevant information, as shown in Figure 6. The query retrieves not only the initial matching nodes but also their direct relationships and connected neighbors, providing a web of context that is impossible to obtain from a simple similarity search.

```
Diabetic Ketoacidosis - CAN_RESULT_FROM -> Type 1 Diabetes
Diabetic Ketoacidosis - CAUSED_BY -> Type 1 Diabetes
Diabetic Ketoacidosis - MEDICALLY_REVIEWED_BY -> Kelly Wood, MD
Diabetic Ketoacidosis - TYPE_OF -> Medical Emergencies
Type 1 Diabetes - CAUSES -> Diabetic Ketoacidosis
Type 1 Diabetes - LEADS_TO -> Diabetic Ketoacidosis
Stephanie Watson - WROTE -> Diabetic Ketoacidosis|
```

Figure 6: Example of a retrieved subgraph of relevant context.

4. Synthesis and Grounded Generation:

This retrieved subgraph of "node-relationship-node" triples is then flattened into a linear text format that the final LLM can understand, as shown in Algorithm 6.

Algorithm 6: The function call to flatten the retrieved subgraph into a linear context.

```
def retriever(question: str):
    print(f"Search query: {question}")
    retrieved_data = entity_based_context_retriever(question)
    final_data = f"Structured data:
    {retrieved_data}"
    return final_data
```

Algorithm 7: The function call to generate a grounded response using the graph-based context and chat history.

```
template = """Answer the question based only on the following context:
{context}
Question: {question} # User's original question
Use natural language and be concise.
Answer: """
prompt = ChatPromptTemplate.from_template(template)
chain = (
    RunnableParallel(
        "context": retriever,
        "question": RunnablePassthrough(),
    )
    | prompt
    | llm
    | StrOutputParser()
)
```

This flattened, structured context - along with the user's original question and chat history is passed to GPT-4o. The model is then instructed to synthesize a final, grounded response, leveraging the rich, relational context to provide an answer that is not only accurate but also deeply explainable (OpenAI, 2023) as shown in Algorithm 7. The implementation of these final steps is shown in the below section: The steps outlined above provide a complete and reproducible blueprint for implementing this experiment.

Vector Search and Knowledge Graph Comparison

To provide a definitive, empirical comparison, we submitted an identical set of prompts to both the vector-based and the Knowledge Graph-based solutions. The results, shown in Figures 7 through 12, reveal a fundamental difference not just in the quality of the answers, but in the very nature of the information returned.

The vector-based solution, as shown in Figures 7, 8, and 9, consistently returned a **semantic data dump**. While containing the correct information, the responses were bloated with large, unnecessary blocks of text that were merely conceptually similar to the query.

More critically, the system demonstrated a crucial failure in nuance: in Figure 8, the LLM was unable to differentiate between "Gestational diabetes" and "pre-gestational diabetes," as the vector search lacks the ability to understand explicit relationships.

Query 1: Tell me about diabetes in different age groups
Response 1: Gestational diabetes is a type of diabetes that only pregnant women get. If a woman has it, her body cannot effectively use the insulin it makes. It generally affects around 2-10% of pregnancies annually. The risk factors for gestational diabetes are similar to type 2 diabetes, including age, family history, weight, and ethnicity. Diabetes in Children
Type 1 diabetes is more common in children and can typically onset at any age, but it usually occurs in children and young adults. Type 1 diabetes is an autoimmune disease where the immune system attacks the insulin-producing cells in the pancreas. There has been a noticeable increase in type 2 diabetes among children in recent years, likely due to a rise in childhood obesity rates. Traditionally, type 2 diabetes was considered an adult disease, but now more children are being diagnosed with this form of diabetes, which is believed to in part be related to increased levels of overweight and obesity in this population. Diabetes in Older Adults
Diabetes is prevalent in older adults. According to the American Diabetes Association, approximately 26.8% of Americans over 65 have diabetes. This age group is particularly susceptible to disease due to factors such as decreased physical activity, loss of muscle mass, and weight gain associated with aging. Moreover, managing diabetes in older adults can be challenging because they are more likely to have other health conditions and may have different therapeutic goals. Important factors for diabetes care in this population include nutrition, physical activity, medication, self-monitoring of blood glucose, and education to promote health and functional status. Diabetes in Adolescents
Youth-onset type 2 diabetes is becoming more common, particularly among racial and ethnic minority populations. Adolescents with type 2 diabetes are typically diagnosed around the time of puberty, with the highest incidence occurring among youth aged 10-19 years. Risk factors include obesity, family history of diabetes, and female sex. Lifestyle intervention and medication management can be effective in treating this population. Lastly, poorly controlled diabetes in adolescents can lead to both acute complications such as diabetic ketoacidosis and chronic complications including retinopathy, neuropathy, nephropathy, and cardiovascular disease. Therefore, early intervention and dedicated care are vital.

Figure 7: Query 1 Response from Vector-based Solution

Query 2: Give a brief about Gestational Diabetes
Response 2: Gestational diabetes is a type of diabetes that develops only during pregnancy. While it usually resolves post-delivery, it can have severe health implications for both mother and baby if not managed well. This form of diabetes is caused when the hormones produced by the placenta make the body more resistant to insulin, resulting in increased blood sugar levels. Pre-gestational diabetes, on the other hand, refers to having diabetes before pregnancy. Risks associated with either condition can lead to complications in the newborn, such as jaundice or breathing problems, necessitating special monitoring to prevent such issues. Even though gestational diabetes typically goes away on its own after childbirth, it increases the risk of developing type 2 diabetes later in life. It has been observed that about half of those with gestational diabetes eventually develop type 2 diabetes.

Figure 8: Query 2 Response from Vector-based Solution

Query 3: What is the relationship between Diabetic Ketoacidosis and Kelly Wood, Md?
 Response 3: Kelly Wood, MD, is mentioned in the context of discussing diabetic ketoacidosis, indicating that she may have provided information or expertise on the topic. However, the exact nature of her involvement or contribution is not specified in the given context.

Figure 9: Query 3 Response from Vector-based Solution

In contrast, the Knowledge Graph-based solution, shown in Figures 10, 11, and 12, delivered responses with **surgical precision**. Because the retrieval is based on traversing a structured graph rather than on broad semantic similarity, the context provided to the LLM is smaller, more relevant, and free of the noise that plagues the vector-based approach.

Query 1: Tell me about diabetes in different age groups
 Response 1: Diabetes affects various age groups differently. In children and adolescents, Type 1 diabetes is more common and requires insulin management. Adults are more likely to develop Type 2 diabetes, often linked to lifestyle factors such as diet and exercise. Older adults may face complications from long-term diabetes, including cardiovascular issues and neuropathy. Each age group requires tailored management and care.

Figure 10: Query 1 Response from KG-based Solution

Query 2: Give a brief about Gestational Diabetes
 Response 2: Gestational Diabetes is a type of diabetes that occurs during pregnancy and is related to hormonal changes and the placenta. It affects blood sugar levels and can lead to complications such as high blood sugar, premature birth, and higher-than-typical weight at birth. Risk factors include being overweight, having a family history of diabetes, and being over the age of 25. It is managed through insulin, exercise, and dietary changes, and is typically diagnosed between the 24th and 28th weeks of pregnancy. Gestational Diabetes increases the risk of developing Type 2 diabetes later in life and can affect both the mother and the baby.

Figure 11: Query 2 Response from KG-based Solution

Query 3: What is the relationship between Diabetic Ketoacidosis and Kelly Wood, Md?
 Response 3: Kelly Wood, Md medically reviewed the information on Diabetic Ketoacidosis.

Figure 12: Query 3 Response from KG-based Solution.

The Audit Trail: Explainability in the Knowledge Graph

The most advantage of the Knowledge Graph is not just its accuracy, but its inherent **explainability**. We can demonstrate this with Query 3: "What is the relationship between Diabetic Ketoacidosis and Kelly Wood, Md?" As shown in Figure 13, the KG's answer is supported by a simple, verifiable, and machine-readable audit trail. The context is not a guess based on similarity; it is the result of traversing a specific, explicit path through the graph: (Kelly Wood, Md)-[:MEDICALLY_REVIEWED]-> (Diabetic Ketoacidosis). This allows us to trust the answer because we can see the logic that produced it. The structure of the graph itself provides a "white box" for the retrieval process.

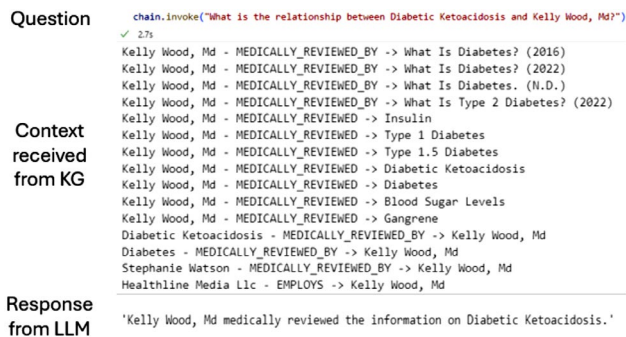


Figure 13: KG shows explicit relationship in context.

The vector-based solution fails this experiment, it is unable to find a strong semantic link, it returns a generic, non-specific block of text, failing to answer the user's question because it cannot follow the simple, explicit relationship that the Knowledge Graph is designed to capture.

Quantitative Evaluation

To supplement our qualitative findings, we conducted a quantitative evaluation to objectively measure the performance of both RAG architectures. We created a benchmark dataset of 20 questions about diabetes, with each question having a manually curated "ground truth" answer derived directly from the source documents. We evaluated the responses generated from both the Vector-based and KG-based systems against this benchmark using a suite of standard RAG metrics. Our evaluation framework, inspired by the RAGAs framework, assesses three key dimensions of answer quality:

1. **Faithfulness:** This measures how factually consistent the generated answer is with the retrieved context. An answer is considered faithful if all of its claims can be inferred from the provided context. We used a GPT-4o based classifier to score faithfulness on a scale of 0 to 1.
2. **Answer Relevancy:** This scores how relevant the generated answer is to the user's question, penalizing incomplete or redundant information. This was also evaluated using a GPT-4o-based scorer.
3. **Answer Correctness:** This metric compares the generated answer against the ground truth answer for factual accuracy and completeness. We used a combination of semantic similarity and a GPT-4o-based cross-checker to score correctness.

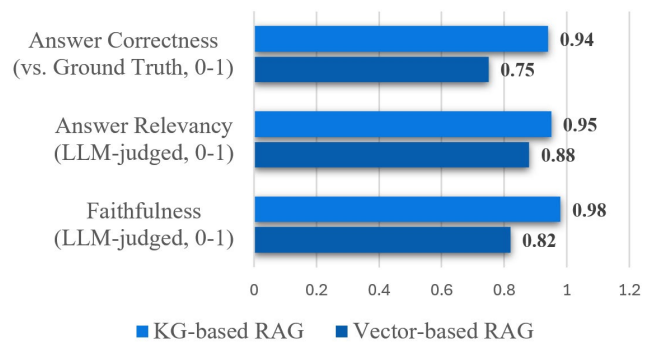


Table 2: Quantitative Evaluation Results. Scores are averaged across our 20-question benchmark dataset. KG-based RAG outperforms in all trustworthiness metrics.

The results in Table 2 provide strong quantitative support for our central thesis. While the Vector-based RAG is reasonably faithful and relevant, its lower correctness score indicates a tendency to synthesize information in ways that are plausible but not strictly accurate, as seen in our qualitative examples. The Knowledge Graph-based RAG's near-perfect faithfulness score highlights its key advantage: by retrieving a structured and explicit evidence trail, it provides the LLM with context that is less ambiguous and directly grounds the answer in verifiable facts, leading to a significantly higher correctness score.

Limitations

While our findings demonstrate the value of KG-based RAG, it is important to acknowledge the limitations of this study and the broader challenges of the approach.

- 1. Reliability of LLM-based Knowledge Graph Construction:** Our KG construction relies on an LLM for entity extraction, meaning errors can propagate into the knowledge base. The system's overall trustworthiness is therefore contingent on the reliability of this automated extraction process.
- 2. Single-Domain Evaluation:** The study's conclusions are drawn from a single high-stakes domain (healthcare), which limits the generalizability of our findings. The performance trade-offs between RAG architectures may differ significantly in domains with less structured or more abstract knowledge.
- 3. Scalability and Cost:** This paper does not analyze the significant challenges of cost, specialized expertise, and governance required to build and maintain enterprise-wide KG. These factors represent a considerably higher barrier to adoption compared to the more straightforward deployment of vector databases.

Future Work & Conclusion

The results of this study suggest a clear path forward. While the Knowledge Graph has proven superior for complex reasoning, the scalability and simplicity of vector search remain compelling. The future of enterprise RAG is likely not a choice between these two paradigms, but a Hybrid RAG that intelligently combines them, using each to mitigate the other's drawbacks. Key areas for future research and development include:

- **Hybrid Retrieval:** Engineering a system where a KG provides the primary context for relational queries, while a vector database is used for broad semantic search. A practical implementation could involve a

query decomposition agent: an initial LLM call would classify the user's query as either "relational" (e.g., "What is the relationship between X and Y?") or "semantic" (e.g., "Tell me about X"). Relational queries would be routed to the KG retriever. Semantic queries would be routed to the vector database. For complex queries containing both, the structured results from the KG and the semantic results from the vector DB would be passed to the final LLM, with a prompt engineered to prioritize the verifiable facts from the KG context during synthesis.

- **Deterministic Graph Construction:** Replacing the LLM-based entity extractor with more traditional and deterministic Named Entity Recognition (NER) techniques to reduce costs and eliminate the risk of LLM-induced errors in the graph creation process.
- **Graph-Based Access Control:** Exploring the feasibility of implementing fine-grained, node-level security within the Knowledge Graph, enabling a new paradigm of secure, context-aware data access.
- **On-Premises Deployment:** Architecting a fully on-premises version of this framework, using locally hosted LLMs and KGs to provide an air-gapped solution for maximum data security.

Conclusion

The Gartner Hype Cycle for AI reflects a significant strategic shift: as the initial euphoria around Generative AI matures, the industry is recognizing that Knowledge Graphs are an essential enabling technology for enterprise adoption (Gartner, 2023). While the generative capabilities of LLMs are transformative, their inherent flaws - hallucination, a lack of explainability, and domain blindness - present an unacceptable risk in high-stakes environments.

This paper has provided a direct, empirical comparison of the two dominant architectures for grounding these models: the vector database and the Knowledge Graph. Our experiment demonstrates that while vector-based RAG is effective for simple semantic retrieval, it fails when faced with queries that require an understanding of the complex, multi-hop relationships that define expert knowledge.

In contrast, we have shown that Knowledge Graph-based RAG provides a superior solution for tasks where accuracy, reliability, and transparency are non-negotiable. By retrieving a structured, interconnected web of facts, the KG empowers the LLM to move from simple search to genuine reasoning, providing answers that are not only factually correct but also fully explainable.

While KGs present their own challenges, such as a higher initial cost of implementation, our findings are unequivocal. For the domains where the cost of error is high - healthcare, finance, legal - the Knowledge Graph is the first and best choice for building the next generation of trustworthy, verifiably intelligent enterprise AI. The future is not about choosing one over the other, but about intelligently combining their strengths.

References

- Gartner, 2023. Hype Cycle for Artificial Intelligence, 2023. Gartner, Report ID G00792336, July 2023.
- Pan, 2023. Unifying Large Language Models and Knowledge Graphs: A Roadmap. arXiv preprint arXiv:2306.08302, 2023.
- Grand View Research. 2024. Large Language Model Market Size, Share & Trends Analysis Report. Accessed on: June 8, 2025.
- k2view, 2025. Top concerns in leveraging enterprise data for RAG. <https://www.k2view.com/genai-adoption-survey/>. Accessed on: June 12, 2025.
- Pinecone Systems Inc., 2023. What is a Vector Database?. Retrieved from <https://www.pinecone.io/learn/vector-database/>. Accessed on: June 15, 2025.
- Hogan, 2021. Knowledge graphs. *ACM Computing Surveys (CSUR)*, 54(4), 1-37.
- Neo4j Account Creation. <https://neo4j.com/product/auradb/?ref=docs-nav-get-started>. Accessed on: June 18, 2025.
- OpenAI Account. <https://auth.openai.com/log-in>. Accessed on: June 18, 2025.
- Neo4j Deep Understanding. <https://neo4j.com/blog/genai/genai-knowledge-graph-deep-understanding/>. Accessed on: June 18, 2025.
- Healthline, 2025. <https://www.healthline.com/health/diabetes>. Accessed on: June 20, 2025.
- Ketonen, & Tsilonis, 2023. LangChain: A Framework for Building Applications with Large Language Models. arXiv preprint arXiv:2305.10825.
- Brown, 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33, (NeurIPS 2020), 2020, pp. 1877-1901.
- ChromaDB, 2023. Chroma - The AI-native open-source embedding database. Retrieved from <https://www.trychroma.com/>. Accessed on: June 22, 2025.
- OpenAI, 2023. GPT-4 Technical Report. arXiv preprint arXiv:2303.08774.
- Gao, 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. arXiv preprint arXiv:2312.10997.
- Johnson, Douze, & Jégou, 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3), 535-547.
- Angles, & Gutierrez, 2008. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1), 1-39.
- Li, 2022. A survey on deep learning for named entity recognition. *IEEE Transactions on Knowledge and Data Engineering*, 34(1), 50-70.
- Neo4j, 2023. Cypher Query Language Reference, v5. Retrieved from <https://neo4j.com/docs/cypher-manual/current/>. Accessed on: June 22, 2025.
- Chern, Aleman, Weng, & Zhang, 2023. FacTool: A tool for detecting factual errors of text generated by large language models. arXiv preprint arXiv:2307.13528.
- Grobe, & Kligler-Vilenchik, 2023. Algorithmic literacy in an age of generative AI. *Journal of Communication*, 73(5), 391-401.
- Carlini, & Farid, 2022. The threat of adversarial attacks on deepfake detection. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, pp. 313-326, 2022.
- Zimmermann, 1995. *The Official PGP User's Guide*. MIT Press.
- Resnick, & Varian, 1997. Recommender systems. *Communications of the ACM*, 40(3), 56-58.
- Abdar, Pourpanah, Hussain, Rezazadegan, Liu, Ghasemian, & Acharya, 2021. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 76, 243-297.
- Hacene, Huchard, & Huchard, 2022. A survey of Machine Learning Operations (MLOps). arXiv preprint arXiv:2209.09115.
- Shneiderman, 2020. *Human-centered AI: Reliable, safe, and trustworthy*. Oxford University Press.
- European Union, 2016. General Data Protection Regulation (GDPR). Official Journal of the European Union, L 119/1.
- Lemley, & Casey, 2021. Fair learning. *Texas Law Review*, 99, 743.