BARC

Mastering the Open-Book Test

Why and How Retrieval-Augmented Generation Improves GenAI Outcomes

Authors: Kevin Petrie & Florian Bigelmaier

Publication: October, 2024

Abstract

This research note defines retrieval-augmented generation, or RAG, a type of workflow that prompts GenAI language models with domain-specific data to improve their accuracy. It evaluates three architectural approaches – vector RAG for semantics, relational RAG for database values, and graph RAG for ontologies – as well as hybrid solutions that combine all three approaches.

Research sponsored by:

Vespa

Table of Contents

Management Summary

Generative AI marks the latest stage of analytics innovation, starting with business intelligence in the 1990s and continuing with AI/ML in the 2010s. But as inputs become less structured and automation more prevalent, governance becomes increasingly challenging – particularly with GenAI. Language models can fabricate answers, mishandle sensitive data or create myriad other governance risks.

This is where retrieval-augmented generation (RAG) comes in. When designed and implemented effectively, RAG provides trusted information and context, improving the likelihood that GenAI language models deliver accurate, governed responses. RAG can significantly enhance GenAI's ability to drive customer satisfaction, boost revenue and streamline operations, creating both growth opportunities and cost efficiencies. Additionally, it provides a foundation for Responsible AI by ensuring governed, ethical and civic-minded practices.

We define three different primary approaches to RAG, each tailored to a distinct data type:

- **Vector RAG** interprets semantic meaning through similarity searches of unstructured data. It is particularly useful for text inputs and can complement searches based on keyword matching.
- **Relational RAG** retrieves accurate values, such as prices or financial results, from relational databases. It enables GenAI to generate responses based on a company's transactional data.
- **Graph RAG** interprets complex relationships between entities in a graph database. It helps language models navigate ontologies, for example related to product portfolios, supply chains or biomedical research.

All these options can benefit from the additional function of text, document and file searching based on keyword matching because this helps the language model start with the right source data. Hybrid approaches are common, as text/keyword search, vector, graph and relational RAG are not mutually exclusive. Many implementations integrate two or more of these approaches, enabling broad retrieval, deep exploration and versatile performance by interpreting data from diverse sources.

To deliver results, RAG must be accurate, modular, adaptable, integrated, resilient, efficient and governed. And it must overcome familiar challenges such as technical debt, data quality issues, computational costs and skills gaps, which can hinder effective RAG design and implementation.

To successfully implement RAG, we recommend the following four guiding principles:

- **Start with lower-risk projects.** These help companies learn valuable lessons before scaling to company-wide or customer-facing initiatives.
- **Evaluate RAG approaches based on complexity**. Determine which aspects of accuracy and precision are critical to your use case and select the right combination of technologies accordingly.
- **Embrace hybrid approaches**. Two or three-pronged approaches (like combining vector and relational databases in your RAG application) make things more complex, but they might increase output accuracy.
- **Consider platforms**. Integrated offerings can simplify implementations and reduce operational risk, especially in the context of hybrid approaches.

Introduction

We all know artificial intelligence (AI) has a trust problem. This is especially true for generative AI, which utilizes neural networks to interpret and create content such as text, imagery or video. Probabilistic by nature, GenAI language models do not hesitate to speculate or fabricate answers to compensate for gaps in training data. These frequent "hallucinations," coupled with other governance risks such as mishandling of private data and intellectual property, threaten to make GenAI more of a liability than an asset.

This research note explores an emerging solution to the problem. Retrieval-augmented generation, or RAG, selects and feeds domain-specific data into language models to improve their accuracy. We define RAG, its challenges, benefits, must-have characteristics and common use cases. We then evaluate three architectural approaches – vector RAG for semantics, relational RAG for database values, and graph RAG for ontologies – and provide an example of a hybrid solution that combines all three approaches. Data and AI leaders that read this report will learn the value of RAG and guiding principles to get started.

The rise of GenAI

The GenAI language model first arose in 2017 when researchers at Google introduced a "transformer" architecture[1] that predicts sequences of outputs based on what it learns from sequences of inputs. This architecture relies on an "attention network" whose parameters quantify the meaning and interrelationships of content chunks – paragraphs, pictures, video clips and so on – within a corpus of training content. Once trained on this attention network, the language model moves to the inference stage in which it generates responses to human questions or instructions, also known as prompts. Many GenAI applications also include evaluation models that assess the performance of individual language models, then select the right ones for different tasks.

OpenAI's prominent release of ChatGPT in late 2022 led to an explosion of GenAI innovation among vendors such as Google, Microsoft, Hugging Face and open-source communities for models such as Llama (backed by Meta). Their massive investments create an opportunity for mainstream companies to improve efficiency and gain competitive advantage. By customizing language models to understand domain-specific data, companies of all types can streamline and enhance business functions such as customer service or document processing. The problem: GenAI poses significant risks, especially those that relate to accuracy and governance. How do we ensure language models understand domain specifics and generate trustworthy responses rather than making things up?

How can companies ensure GenAI language models don't make things up?

[1] Vaswani et al. (2017) Attention Is All You Need, https://arxiv.org/abs/1706.03762

Definition of RAG

Retrieval-augmented generation offers a potential solution. RAG is a workflow that retrieves enterprise data and uses it to augment the questions that humans pose to GenAI language models. This is akin to the "open-book" tests we took in school, in which we answered teacher's questions while consulting textbooks. In essence, RAG gives the language model a textbook to help answer the user's question. Designed and implemented well, it provides trusted information and context that increase the likelihood of governed, accurate responses. RAG has emerged as a preferred way to govern GenAI and customize it to address companies' specialized business problems. It typically requires less AI expertise, takes less effort and consumes less compute than fine tuning, another method of customizing language models and improving accuracy. It also minimizes the need for humans to find trusted content and add it to the prompt themselves, also known as a type of "prompt engineering."

In essence, RAG gives the language model a textbook to help answer user questions

Our definition of RAG spans four architectural layers: source data, pipelines, databases and applications (see figure 1).

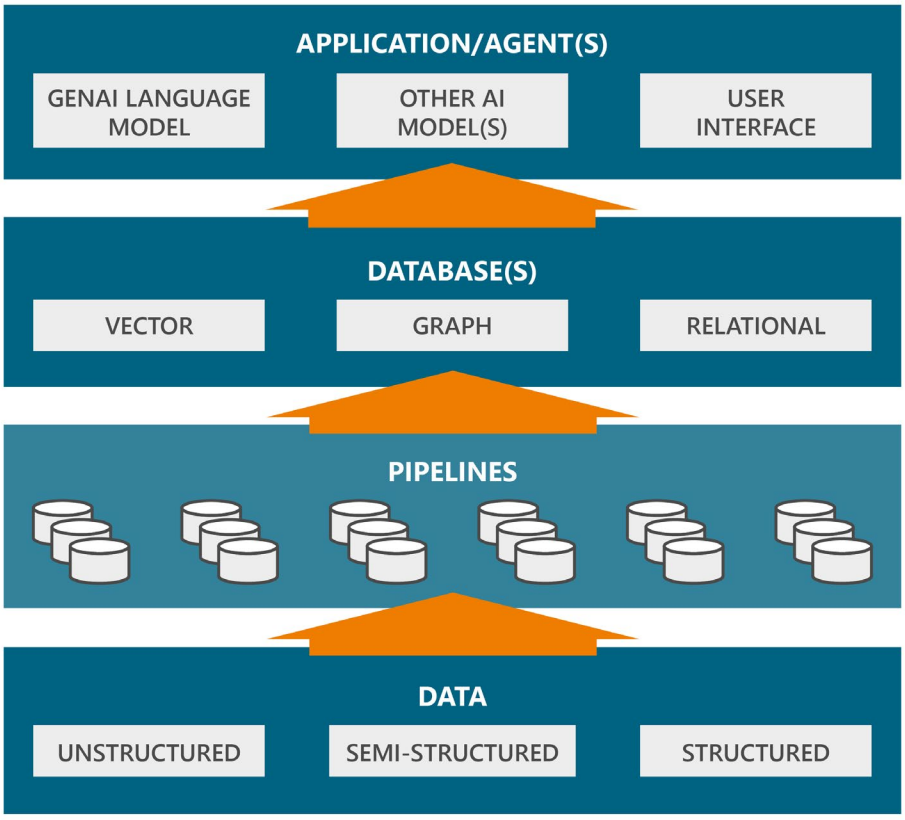

Figure 1: Generic RAG Architecture

- **Data**. Common data sources include structured tables, semi-structured objects such as logs or labeled images, and unstructured objects such as text, video or audio files. Enterprise environments have mountains of such data, mostly unstructured and often untouched by analytics projects.

- **Pipeline**. Using a familiar ETL sequence, pipelines extract these multi-structured data objects, transform them into usable objects such as vector embeddings, then load them and the associated metadata into target databases. Alternatively, an ETLT or ELT pipeline might transform the data within the database, for example by creating the nodes of a graph.
- **Database**. A vector, graph or relational database – or often some combination thereof – organizes and indexes these objects. Databases parse queries, then search and retrieve objects such as vector embeddings, graph nodes/edges or relational tables. For example, a vector database will perform a similarity search of embeddings based on their semantic meaning and inter-relationships.
- **Applications/agent**. When prompted by a user question, the application containing the language model queries the database for relevant data. It injects that data into the user prompt, then delivers the language model's response back to the user through a chatbot or other interface. Applications might evaluate, select and orchestrate multiple types of AI models beyond just GenAI. They might even operate as autonomous "agents" that take action without further human instructions.

RAG architectures span source data, pipelines, databases and applications

AI market evolution

RAG arrives at a time of rising risk for data analytics. We see this trend when we consider the three primary stages of this evolving market: business intelligence, AI/ML and now GenAI. With each new stage, inputs become more complex and outputs become more sophisticated (see figure 2).

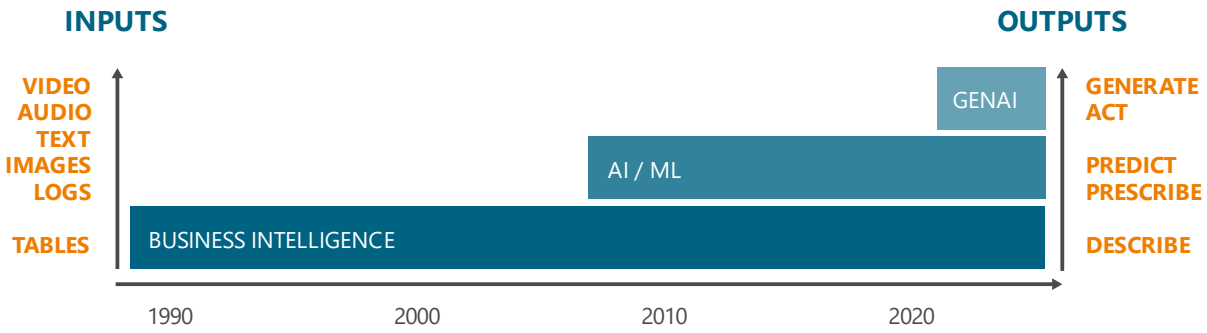

Figure 2: Evolution of Data Analytics

- **Business intelligence**. Since the 1990s, data analysts have measured historical company performance and market trends based on periodic and now real-time assessments of database records. They use BI tools to build reports and dashboards that inform operational business decisions. BI projects like these consume structured data – i.e., tables.
- **AI/ML**. Since the 2010s, data scientists have employed AI to assist decisions and automate actions. They train machine learning models to classify, predict and prescribe future outcomes based on historical patterns. Such models consume tables as well as images, text files or other semi/unstructured inputs.

- **Generative AI**. Data scientists now train and implement GenAI language models that generate content to respond to natural language prompts. In addition, developers build applications that use language models alongside other models to converse with users, make recommendations and take automated action.

Each phase enriches rather than replaces what came before. Many BI tools now include language models that help data analysts query data, visualize outputs, interpret results and so on. And multifaceted applications containing diverse model types now manipulate and consume diverse data types. With each new stage, data analytics has become less structured and more automated, which increases the risk of ungoverned outputs that might damage the business. This raises the need for RAG to assist GenAI-related initiatives.

Challenges

Several familiar challenges stand in the way of effective RAG design and implementation. For years, data teams have accumulated **technical debt** – including manual processes, siloed systems and incompatible tools – by taking shortcuts with individual projects. Data analysts, engineers and stewards continue to struggle with **data quality issues** thanks to duplicative, contradictory or siloed tables and files. This contributes to the larger, related challenge of data and AI governance: companies struggle to implement policies that safeguard privacy, transparency, accountability, compliance and – above all – human control of algorithms.

Companies also have a **skills gap**: their data engineers must learn to manage unstructured data, their data scientists must learn to manage GenAI, and both groups must learn how to collaborate. **Cost-performance** looms as a final challenge: all these stakeholders must collaborate with developers and CloudOps engineers to ensure RAG workloads meet performance requirements without exceeding their compute budget. They might need new tools to get the granular insight they need into resource utilization. Given these various challenges, it's no surprise that most GenAI and RAG projects remain in the planning or pilot stage. Getting to full-scale production takes time, effort and iteration.

Benefits

Designed and implemented well, RAG can enable GenAI to deliver on its intended benefits. GenAI-enabled applications can win, delight and upsell customers: picture the chatbot that turns an irate user into a repeat buyer. They help build revenue streams: picture the conversational smartphone app that opens the door to a new target market. RAG-enabled GenAI initiatives can also reduce costs by streamlining operations, from back-office functions to sophisticated assembly lines. Finally, RAG provides a solid foundation for Responsible AI: that is, governed, ethical and civic-minded AI practices.

Must-have characteristics of RAG workflows

To deliver results, RAG must be accurate, integrated, modular, adaptable, resilient, efficient and governed.

- **Accurate**. An effective RAG workflow improves the accuracy of language model outputs by prompting them with knowledge of domain-specific semantics, ontologies or values – or more likely all three.

- **Integrated**. Integrated platforms and tools simplify implementation and reduce risk. For example, AI databases such as Vespa support keyword matching, semantic search and relational queries in a single offering. Vectara, meanwhile, integrates model workflow tasks within its RAG as a service.

- **Modular**. While integrated, RAG should comprise modular elements in case data and AI teams need to insert, remove, revise or replace them to meet changing business requirements. These modular elements, including datasets, pipelines and models, depend on open APIs and data formats.

- **Adaptable**. Data and AI teams need the flexibility to adapt RAG workflows in a variety of ways: scaling up compute clusters to support a workload burst, tuning a pipeline to optimize performance and so on.

- **Resilient**. RAG must overcome disruptions and adverse conditions to maintain acceptable service levels. It might need automatic failover to respond to server failures, mirrored object stores to avoid data loss, or continuous monitoring to spot slowdowns. Safeguards such as these help these workflows remain available and perform for users.

- **Efficient**. As with other technologies, compute-intensive workloads can lead to unpleasant cost surprises. An efficient RAG workflow helps data teams stay within budget by consuming compute cycles sparingly and monitoring workloads to anticipate and prevent overruns.

- **Governed**. A RAG workflow must enforce governance policies, rules and standards on several levels. It must observe data quality or bias; protect user privacy; audit operations; control policies and access; and provide sufficient transparency to make AI models explainable. Governance also requires expert human oversight of RAG design, deployment and operation.

To deliver results, RAG must be accurate, integrated, modular, adaptable, resilient, efficient and governed

Architectural approaches

Now let's explore three primary architectural approaches: Vector RAG specializes in interpreting semantic meaning, relational RAG retrieves and calculates database values, and graph RAG interprets ontologies (see figure 3).

	VECTOR	RELATIONAL	GRAPH
SEMANTICS	X		
VALUES		X	
ONTOLOGIES			X

Figure 3: RAG Approaches and Their Strengths

All these options can benefit from the additional function of document and file searching based on keyword matching because this helps the language model start with the right source data. Data and AI leaders should evaluate how each approach's distinct methods of organizing data map to their domain-specific needs. In all likelihood, they will combine two or three approaches to achieve the right level of accuracy.

Vector RAG

Vector RAG interprets semantic meaning based on similarity searches of unstructured data, especially chunks of text, sometimes within a vector database. Vector RAG helps when word choice really matters. For example, it might help a GenAI application use nuanced legal language when summarizing court cases for attorneys. Given the popularity of text inputs for GenAI, vector RAG has become the most common RAG approach. For example:

- MyWealthAI bases its GenAI assistant for Australian investors on vector RAG. It uses Vectara's RAG as a service to filter, select and summarize content from resources such as company announcements and research reports.
- Spotify offers natural-language search capabilities to its customers based on vector RAG. It uses Vespa's AI database to index, search and rank vectorized podcast content.

Vector RAG interprets semantic meaning based on similarity searches of unstructured data

A vector RAG workflow involves unstructured data, pipelines and the vector DB; as well as application(s) or agent(s) that contain the language model, other AI model(s) and a user interface (see figure 4).

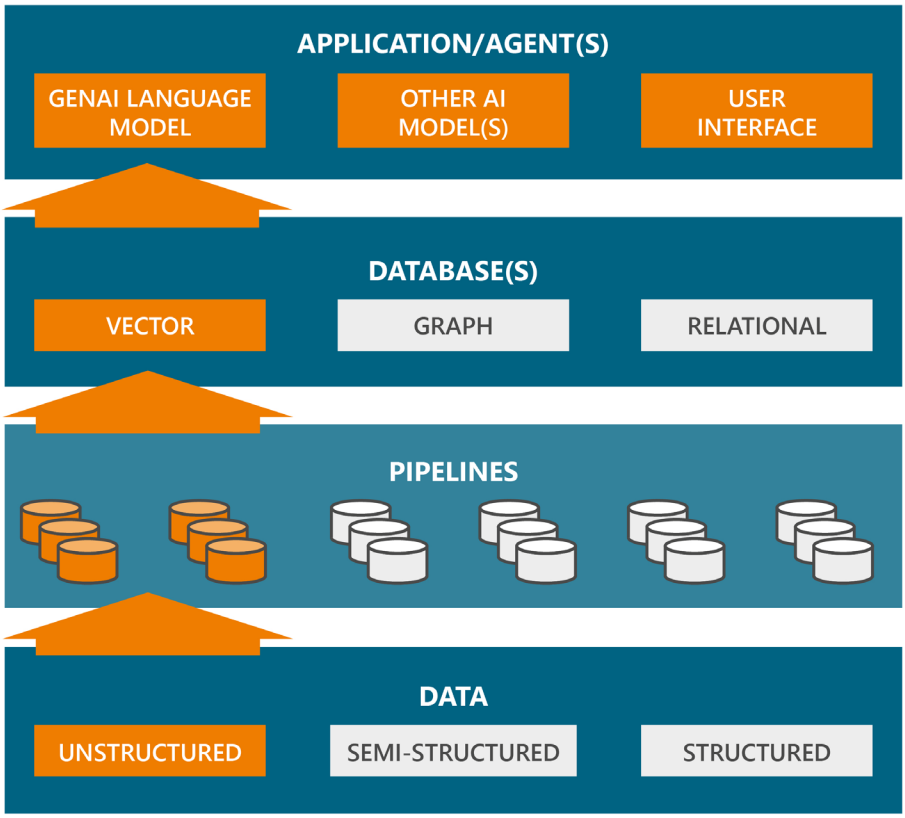

Figure 4: Vector RAG Architecture

- **Data**. Vector RAG relies on unstructured data within one or more object stores. While plain text is the standard, more complex formats such as video and audio have also become popular.
- **Pipelines**. Pipelines chunk text into smaller, manageable pieces and convert those chunks into numerical representations called embeddings. These embeddings capture the semantic meaning and relationships of the chunks in a high-dimensional vector space.
- **Database**. The vector database stores the embeddings in an index along with the chunks themselves. Its main role is to organize the data for efficient retrieval.
- **Applications/Agents**. When a user submits a query, the application converts the query into an embedding. The vector database then searches for source chunks with similar embeddings. The application retrieves the chunks, ranks them and injects the most relevant chunks into the language model's prompt.

The quality of the indexed data determines the completeness, precision and accuracy of the output. Best practices for unstructured data include regularly refreshing data and filtering out irrelevant documents to reduce noise. Developers can also improve output quality by integrating document metadata, applying advanced reranking algorithms or combining keyword and vector searches during retrieval. Business and data/AI experts should oversee the design, implementation and operation of the vector RAG workflow to ensure governed answers.

The pros and cons of vector RAG break out as follows:

- **Pros**. Vector RAG scales efficiently to accommodate large data volumes and user traffic. It also saves compute cost by reducing the number of input tokens and therefore the prompt size.
- **Cons**. Vector RAG does not specialize in handling tables, which often contain the most trusted business facts. Further, semantic similarity is a smart estimation, a kind of heuristic. It does not explicitly manage or hold modeled knowledge, which raises the risk of hallucinations or irrelevant answers for complex domains.

Relational RAG

Relational RAG retrieves and calculates trusted values, such as purchase amounts, product prices or financial results, by querying relational database queries to ensure high levels of accuracy. When customers ask about their bill or insurance subscribers ask about claim status, the language model needs to give them a correct answer – and the correct values often come from a relational database.

Relational RAG retrieves trusted facts and calculates values from a database

Relational RAG helps casual users by increasing access to insight compared with visual data analysis workflows and traditional SQL queries. More and more business intelligence (BI) vendors are including such functionality in their software. Standalone relational RAG solutions are also possible, especially for specific use cases (see figure 5).

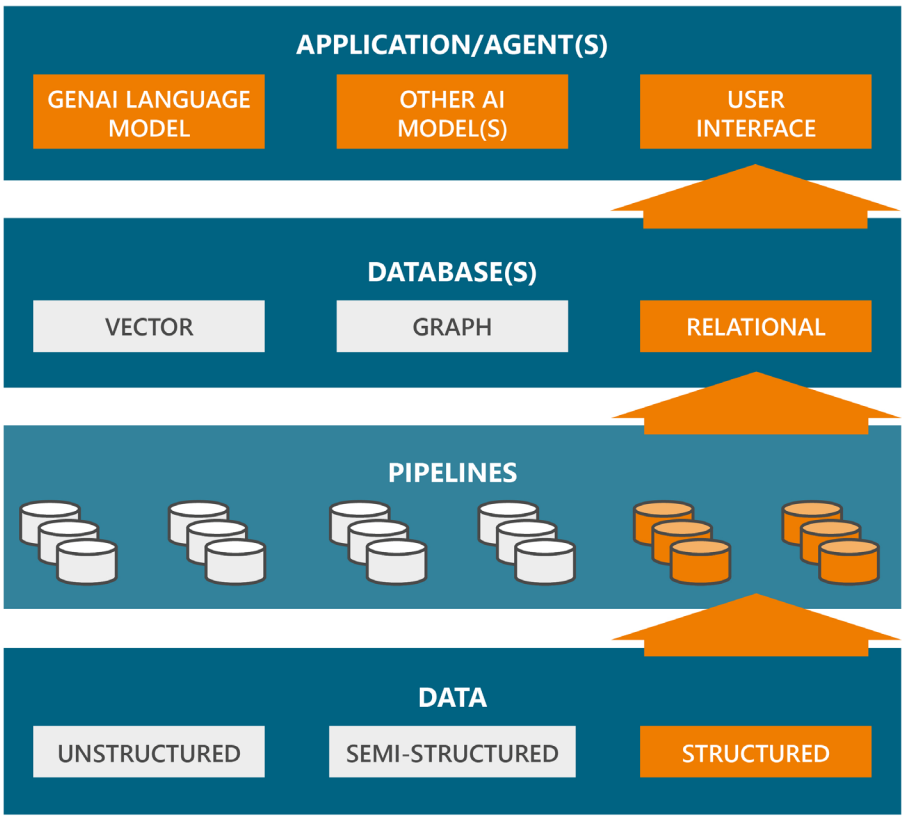

Figure 5: Relational RAG Architecture

- **Data.** Relational RAG begins with well-organized relational data such as transactional records or customer data arranged in tables with clearly defined relationships. This requires data teams to have mature processes and governance practices in place beforehand.
- **Pipelines.** Pipelines prepare data for consumption in a RAG workflow. For example, they might extract, transform, then load it into the database (ETL). Analysts might also calculate relevant metrics in this step. The pipeline also collects and catalogs metadata to assist subsequent steps.
- **Database.** The relational database stores the resulting analytical data, including cleaned operational records and pre-aggregated metrics. Metadata describes the complex data structures and abstracts them into business-friendly terms.
- **Applications/Agents.** The application maps the relevant portions of the user input to a database query language such as SQL. The metadata can also populate a dictionary that supports natural language queries. The application retrieves the data and the language model generates an answer.

In addition, the language model can generate the database query itself as an intermediate step. Architectural variations are possible; for example, the workflow might use a REST API to retrieve structured data instead of querying the database directly.

- **Pros.** Relational RAG offers an established way to answer questions about business performance. It enables casual users to access database values and enables power users to reduce time to insight.
- **Cons.** Research shows that even when investing in (meta)data quality, these systems cannot guarantee correct answers, especially for complex queries. A transparent system that explains its results, combined with human oversight, can mitigate this problem.

Graph RAG

Graph RAG interprets ontologies by modeling domain knowledge to reconcile complex relationships between entities and concepts within a graph database. Graph RAG helps language models interpret a wide array of ontologies, from product portfolios to supply chains or biomedical research. For example, a wholesaler might have clients that need to navigate thousands of categories, products and components. While graph RAG often serves as a basis for specialized or advanced chatbots, it also assists R&D for complex question-and-answer processes.

Graph RAG interprets ontologies by reconciling complex relationships between entities and concepts within a graph database

The graph RAG workflow involves diverse data, pipelines, the graph database and applications/agents (see figure 6).

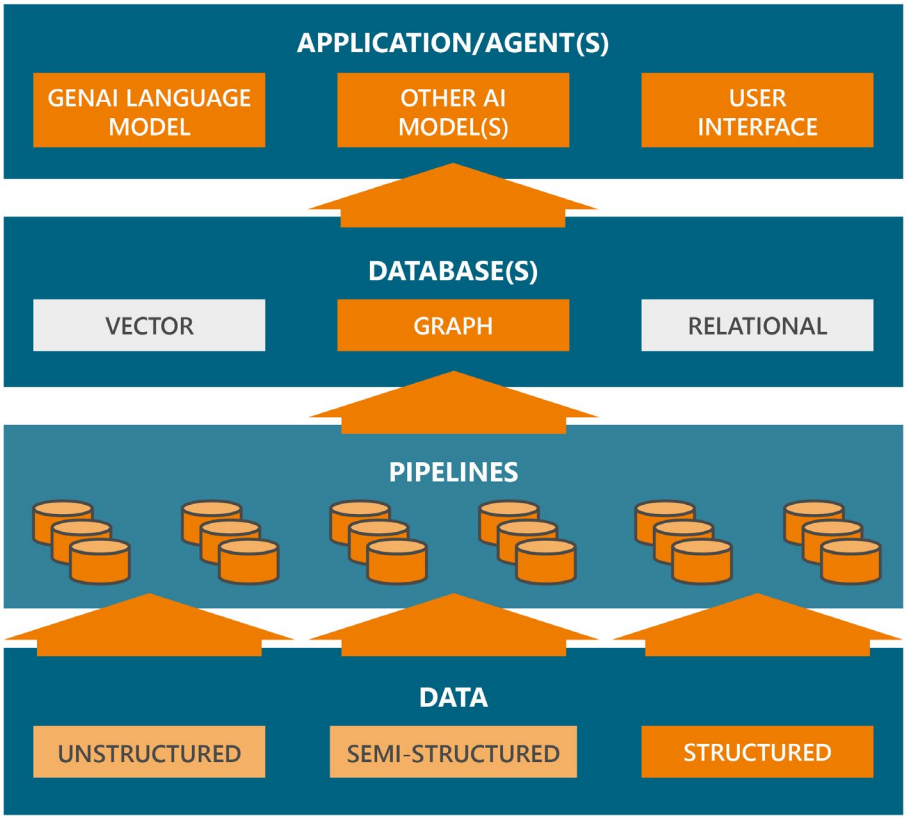

Figure 6: Graph RAG Architecture

- **Data.** Graph RAG applies to all types of data, although structured data is most common and compatible with knowledge graphs. Existing ontologies and taxonomies such as business glossaries need little transformation before they populate a graph database.
- **Pipelines.** Transformation takes place within the pipelines. Data teams might manually create structures that assemble raw data into graphs. They might also apply new AI techniques to automate the extraction of entities (nodes) and relationships (edges) from unstructured data.
- **Database.** The graph structure emphasizes relationships and context that are fundamental for modeling knowledge. Most graph databases also integrate text sources to enrich their representation of data structures. In addition, many master data management (MDM) databases contain knowledge graphs that describe the ontological relationships of business entities such as customers or products.
- **Applications/Agents.** When a user submits a query, the system identifies relevant nodes, relationships or subgraphs that help answer the question. Next, the application prioritizes the results and injects the most relevant data into the prompt. Based on this guidance, the language model generates the final output.

Graph modeling methods have a significant impact on the accuracy of language model outputs. Organizations need to find their individual sweet spot of invested effort and output quality.

Pros and cons break out as follows.

- **Pros.** Graph RAG has the potential to deliver comprehensive answers by uncovering complex relationships and providing control over knowledge from multiple sources. Use cases that require complex reasoning (as in R&D) or that involve elaborate taxonomies (as in healthcare) can benefit from this approach.
- **Cons.** Graph structures might be complex to set up and require extensive effort and knowledge. The compute cost of running a graph RAG application might exceed budgets for high-scale, low-latency implementations.

Hybrid approaches

Vector, relational and graph RAG approaches are not mutually exclusive. Many early RAG implementations combine two or all three approaches, along with document searches based on keyword matching. Such hybrid approaches exploit the complementary strengths of different technologies to produce better results with diverse data sources. By integrating and interpreting diverse data sources, they enable broad retrieval, deep exploration and versatile performance.

For example, consider a logistics company that implements a GenAI application to enhance customer service for various wholesale and retail clients. This GenAI application includes a chatbot and multiple language models, supported by hybrid RAG, for handling questions and answers. A combination of keyword search and vector RAG pulls up the right product documents to assist with questions about product and component specifications. Keyword search focuses on exact matches, while vector RAG excels at finding content that is semantically connected to the question. Graph RAG provides knowledge of the product taxonomy. In addition, relational RAG retrieves database records such as client orders and invoices to ground the conversations in trusted facts.

The GenAI application evaluates and selects an appropriate language model to generate well-grounded responses based on each of these inputs. The logistics company's clients get the trusted and contextual responses they need to handle a variety of situations. While the hybrid approach increases implementation complexity, the benefit of higher customer retention outweighs the cost.

Guiding principles

As we remember from school, open-book tests pose more challenges than one might expect. You have a limited amount of time to pull up just the right facts and analyze them in a coherent fashion. To master the test, a student needs a contextual understanding of the subject domain. In the data world, RAG faces its own challenges: technical debt, data quality, skills gaps and cost-performance. RAG must be accurate, integrated, modular, adaptable, resilient, efficient and governed to master the test.

In this report, we presented three complementary architectural approaches – vector, relational and graph – for achieving these must-have characteristics. We recommend that data and AI leaders apply the following guiding principles:

- **Start with lower-risk projects**. Departmental projects that focus on internal users, for example as part of efficiency initiatives, pose lower risk to the business. They also might require lower levels of accuracy because employees can apply their own expertise to govern GenAI inputs and outputs. Initial projects like these help companies learn valuable lessons before scaling to company-wide or customer-facing initiatives.

- **Evaluate RAG approaches based on complexity**. Ask what dimension of accuracy and precision matters most to your initial use case and project. If you require domain-specific word choice based on semantic search, then vector RAG can help. If you require careful handling of complex ontologies, then graph RAG can help. Relational RAG, meanwhile, delivers trusted facts and figures from database tables – which might prove critical for all your use cases.

- **Embrace hybrid approaches**. As you move to more sophisticated use cases, more complex domains and higher-risk projects, consider adopting two or even three-pronged RAG approaches to deliver the highest accuracy possible. While hybrid approaches can complicate your implementations, they bring together insights from multiple sources and compensate for one another's limitations.

- **Consider platforms that support hybrid approaches**. Evaluate how consolidated platforms can simplify your implementation of hybrid approaches. For example, AI databases such as Vespa support vector and relational RAG approaches as well as keyword search. Vectara, meanwhile, provides a RAG service that spans the end-to-end workflow. As described in our must-have characteristics section, integrated offerings can simplify implementations and reduce operational risk.

About BARC

BARC (Business Application Research Center) is one of Europe's leading analyst firms for business software, focusing on the areas of data, business intelligence (BI) and analytics, enterprise content management (ECM), customer relationship management (CRM) and enterprise resource planning (ERP). Our passion is to help organizations become digital companies of tomorrow. We do this by using technology to rethink the world, trusting databased decisions and optimizing and digitalizing processes. It's about finding the right tools and using them in a way that gives your company the best possible advantage. This unique blend of knowledge, exchange of information and independence distinguishes our services in the areas of research, events and consulting.

Research

BARC user surveys, software tests and analyst assessments in blogs and research notes give you the confidence to make the right decisions. Our independent research gets to the heart of market developments, evaluates software and providers thoroughly and gives you valuable ideas on how to turn data, analytics and AI into added value and successfully transform your business.

Consulting

The BARC Consulting practice is entirely focused on translating your company's requirements into future-proof decisions. The holistic advice we provide will help you successfully implement your data & analytics strategy and culture as well as your architecture and technology. Our goal is not to stay for the long haul. BARC's research and experience-founded expert input sets organizations on the road to the successful use of data & analytics, from strategy to optimized data-driven business processes.

Events

Leading minds and companies come together at our events. BARC conferences, seminars, roundtable meetups and online webinars provide more than 10,000 participants each year with information, inspiration and interactivity. By exchanging ideas with peers and learning about trends and market developments, you gain new impetus for your business.

BARC

About Vespa.ai

Contact info

Vespa.ai Norway AS
Prinsens Gate 49
Trondheim 7011
Norway

vespa.ai

Mail: info@vespa.ai

Vespa.ai is the developer of Vespa, a platform for building and running real-time AI applications, such as search, recommendation, personalization, and conversational AI. The platform supports retrieval-augmented generation (RAG), vector databases, large language models (LLM), and machine learning. It efficiently manages data, inference, and logic for applications handling large datasets and high concurrent query rates. Vespa combines multiple data types – vectors, text search, structured, and unstructured data – for precise hybrid search, ranking, and inference.

Available as both a managed service and open source, Vespa is designed for low latency and high scalability. It easily handles over 100K concurrent queries per second and is trusted by companies such as Spotify, Wix, and Yahoo for large-scale deployments.

Vespa's distributed architecture ensures scalability and fault tolerance by distributing data, queries, and models across multiple nodes. This allows it to scale horizontally and vertically, increasing capacity and performance without needing specialized processors like GPUs. By performing computations where data is stored, Vespa reduces data transfer costs, latency and supports corporate privacy and security policies.

The platform is highly customizable, with APIs and SDKs for easy integration. Its robust metrics and logging capabilities help monitor performance and resolve bottlenecks, ensuring reliable, efficient operation as data and query rates grow.

BARC

Data Decisions. Built on BARC.

www.barc.com

Germany
BARC GmbH
Berliner Platz 7
D-97080 Würzburg
+49 931 880651-0

Austria
BARC GmbH
Hirschstettner Straße 19 / I / IS314
A-1220 Wien
+43 660 6366870

Switzerland
BARC Schweiz GmbH
Täfernstr. 22a
CH-5405 Baden-Dättwil
+41 56 470 94 34

United States
BARC US
13463 Falls Drive
Broomfield, CO 80020
USA