Vespa.ai

Enabling Generative AI in the Enterprise with Retrieval Augmented Generation (RAG)

A Guide for Managers

We Make AI Work

Vespa.ai

We are a platform for building and running real-time AI-driven applications for search, recommendation, personalization, and RAG. It enables enterprise-wide AI deployment by efficiently managing data, inference, and logic, handling large data volumes and over 100K queries per second.

Document
Enabling Generative AI in the Enterprise with Retrieval Augmented Generation (RAG)

01.01.2025

Contents

Introduction

Much has been written about the impact of generative AI on business transformation, so it's unnecessary to repeat it here. Instead, this document explains the key technologies and concepts required to implement generative AI in a business, offering a guide for making informed, actionable decisions.

Large language models (LLM) have taken the world by storm. Part of the appeal is the ease with which products like ChatGPT carry out tasks such as enriching search outcomes with text answers instead of listing websites, creating new content, or conducting research. ChatGPT can do this straight out of the gate because it has been trained on public information on the Internet.

But what if you want to use an LLM on some problem involving real-time or private company data that is not on the public internet? You would need to supply that data to the LLM to solve the problem at hand. This is the purpose of retrieval-augmented generation (RAG). RAG combines the power of LLMs with your organization's unique data sources, enabling the model to retrieve and integrate real-time or private information. This allows you to generate highly relevant, context-rich responses appropriate for your business using your data—whether for customer support, knowledge retrieval, company-specific insights or any other generative AI initiative.

What is RAG?

Retrieval-augmented generation (RAG) allows generative AI models to securely access and use a company's private data to solve business problems. Understanding RAG is essential for any organization leveraging or planning to leverage generative AI.

Unlike traditional enterprise decision-making, such as Business Intelligence (BI), which consolidates operational data into a data warehouse for structured analysis and reporting, RAG enhances efficiency by dynamically retrieving only the most relevant information from vast corporate data sources. This approach ensures that generative AI responses are accurate, timely, and informed by the latest context, making it especially valuable for applications that depend on rapidly changing data.

RAG operates in two steps:

- **Retrieval**: Search vast data collections for information relevant to the user's question or task.
- **Generation**: Use this targeted information to generate an accurate, context-aware response using an LLM.

Killer applications for RAG span multiple sectors, including recommendation engines, personalized experiences, enterprise search, customer support, healthcare image analysis, legal document analysis – any use case that can be automated or improved with generative AI.

The RAG Workflow

Embedding

A RAG workflow is the online end-to-end process from understanding the task, through information retrieval to the LLM generating relevant and accurate responses. Here's a breakdown of the steps in a typical RAG workflow:

1. **Query Processing:** A user poses a question or task. This may be preprocessed to improve retrieval accuracy by refining and focusing on keywords or intent.
2. **Contextual Retrieval:** The retrieval model searches a knowledge base for relevant content by understanding the query's meaning. This approach finds conceptually related documents, even without exact keyword matches. For example, in eCommerce, searching for a "stylish work backpack" might yield high-quality leather bags or durable laptop backpacks that fit the user's intent, even if those words aren't used.
3. **Relevance Ranking:** Retrieved documents are ranked by relevance, using machine learning algorithms, based on similarity, context and user preferences and behavior.
4. **Knowledge Aggregation:** The most relevant results are selected and aggregated, preparing them to be passed to the generative model.
5. **Generative Response:** Using the selected context, a LLM (such as GPT) creates a natural language response.

Embedding is a crucial process that transforms text, images, or other data types into numerical representations (vectors) that capture the semantic meaning of the content. This allows the AI to understand context and similarity, retrieve relevant information and generate accurate responses. Complex documents are embedded before the RAG process begins rather than in real-time. This pre-processing approach helps optimize workflow speed and efficiency, especially with large or complex document sets. This is typical for applications with large amounts of static data, such as PDF documents in a bank.

The timescales for pre-embedding depend on the document's significance. For time-sensitive applications, such as financial contracts or frequently updated product catalogs, new documents are embedded as they are created or uploaded. This ensures that the latest information is immediately available for retrieval, enabling the RAG workflow to respond with the most current data.

Embedding

Embedding is a crucial process that transforms text, images, or other data types into numerical representations (vectors) that capture the semantic meaning of the content. This allows the AI to understand context and similarity, retrieve relevant information and generate accurate responses. Complex documents are embedded before the RAG process begins rather than in real-time. Pre-processing helps optimize workflow speed and efficiency, especially with large or complex document sets. This is typical for applications with large amounts of static data, such as PDF documents in a bank.

The timescales for pre-embedding depend on the document's significance. For time-sensitive applications, such as financial contracts or frequently updated product catalogs, new documents are embedded as they are created or uploaded. This ensures that the latest information is immediately available for retrieval, enabling the RAG workflow to respond with the most current data.

For less urgent applications, embedding is often done in batches—either daily, hourly, or at another regular interval. This batch approach is efficient when dealing with a large volume of documents, as it reduces computational strain by processing multiple documents at once. Batch embedding is suitable for platforms with predictable content updates, such as nightly updates for document archives or knowledge bases.

An embedding and retrieval model gaining much attention is ColPali. ColPali is an open-source model that simplifies and enhances information retrieval from complex, visually rich documents, including PDFs. It enhances document retrieval by embedding entire rendered documents, including visual elements, into vector representations optimized for LLMs. ColPali eliminates complex preprocessing, preserves visual context, and streamlines the RAG pipeline by treating documents as visual entities rather than text.

However, embedding is not always necessary. In text-heavy applications, like product reviews in eCommerce or Know Your Customer (KYC) in financial services, BM25 (Best Matching 25) can be a more effective, efficient and vastly simpler alternative. BM25 ranks documents by relevance to a given query, utilizing term frequency (how often a term appears within a document) and inverse document frequency (the terms uniqueness across documents) to assess relevance without the need for embedding.

Vector Search

While data retrieval can be from various sources, RAG often uses data stored as embeddings in a vector database. A vector database is a specialized database that stores data in a dimensional form, giving structure to unstructured data such as images, text, and audio. Vectors show the semantic similarity between pieces of data. This is essential for tasks where you need to understand the meaning or similarity between items, such as recommendation engines, semantic search, and matching tasks, where you want to retrieve items most similar to a given input.

This is referred to as a dense vector search, where every position in the vector has a value, and the values are derived from the entire context of words or sentences. For example, on an eCommerce website with dense vector search, a search for "comfortable summer shoes for the beach," converts the query into a set of numbers that capture the overall meaning of the phrase, rather than just the individual words. Instead of just looking for items with the exact keywords "comfortable," "summer," or "shoes," the dense vector search matches the query with product descriptions that have a similar meaning. This presents results such as:

- "Lightweight beach sandals"
- "Breathable sneakers for warm weather"
- "Comfy slip-ons ideal for outdoor summer use"

Even though these results may not contain the exact search words, the dense vector search understands the context—something comfortable, summer-appropriate, and suited for the beach. This allows it to return relevant options that might be missed if the engine only relied on exact keyword matching. In contrast, a keyword-based or sparse vector search would show results only for products containing the exact words "comfortable," "summer," and "shoes," which could be less helpful.

Vector Search (Continued)

Vector searches help find similar meanings but sometimes lack accuracy because they rely on semantic similarity rather than exact term matching. This can lead to contextually close results that don't precisely match the user's intent. This issue often arises due to nuanced meanings in language that the model might misinterpret.

For example, consider a search on an eCommerce site for "formal black dress shoes." A vector search might retrieve results semantically similar to the idea of "formal shoes" or "black shoes" but miss the exact type of shoe the customer wants. As a result, the search may surface:

- Black sneakers or loafers (similar color and general category but not formal dress shoes)
- Formal brown dress shoes (matching "formal" and "dress shoes" but missing the color specification)
- Men's formal dress shoes (even if the user is looking for women's shoes, since the model might focus on general terms rather than fine details)

This inaccuracy occurs because vector models prioritize capturing conceptual similarity rather than exact matches on specific attributes like color, style, or intended use. Vector search excels in finding conceptually related items, but it can lack the granularity needed for highly accurate results in cases where precise details are essential.

Combining vector search with traditional keyword-based approaches, like BM25, can help mitigate this by balancing semantic understanding with exact term matching, especially when exact product details or attributes are critical.

Hybrid Search

Robust retrieval spans any data type – structured, unstructured, vectors and text – across multiple data sources. It provides tools for searching for exact phrases, ranking results by relevance, summarizing content dynamically, and letting users refine their search with filters. Limiting retrieval to vector search may overlook important data and massively impact the outcome of the generative AI model.

Hybrid search combines traditional keyword-based retrieval with semantic vector search to deliver more accurate and relevant results by leveraging both exact term matching and contextual understanding. Businesses achieve more precise results by integrating traditional techniques like BM25, which ranks documents based on keyword frequency, with advanced neural ranking models that learn from data patterns. Studies, such as Perspectives on R in RAG, have demonstrated that these hybrid models outperform using either method alone, particularly when dealing with new or unfamiliar topics.

In real-world search systems, ranking results often depend on real-time information like whether an item is in stock, its popularity, or other business rules. These factors are hard to fit into a basic vector similarity calculation, making it necessary to use additional ranking methods beyond vectors. Combining different data sources—semantic vectors, exact text matches, and structured metadata—achieves the accuracy needed for real-world applications. Feeding all these inputs into a decision-making model ensures you ultimately surface the most relevant and accurate information.

Addressing Latency and Reducing Cost

While generative AI is gaining traction, mainstream deployment in business operations is yet to happen. For example, an MIT Technology Review survey of 300 senior executives revealed that while 76% of companies experimented with generative AI in 2023, only 9% had adopted it widely. Most organizations plan to expand its use indicating current limited deployment.

Transitioning generative AI “experiments” from concept to full enterprise deployment presents substantial challenges. Poorly designed run-time deployment can result in runaway costs or poor search capabilities, undermining generative AI's potential advantages. For instance, integrating dense vector search with traditional search methods can enhance accuracy but often brings increased latency, higher costs, and scalability issues. Effectively addressing these challenges calls for thoughtful architectural planning and a deep understanding of the search system’s core infrastructure to ensure that it can deliver high performance at scale without compromising efficiency.

Here are some techniques for optimizing RAG workflows, allowing businesses to run large-scale applications without breaking the bank.

- **Optimized Query Execution**: Rapid handling of queries through efficient execution, parallel processing, and smart caching strategies.
- **Dynamic Auto-Scaling**: This automatically adjusts resources in real time based on query complexity to ensure efficient resource use and performance during peak traffic. Horizontal scaling adds nodes for higher query volume, and vertical scaling enhances node capacity.
- **GPU Acceleration:** Utilizes GPU power for high-dimensional embedding similarity calculations, significantly boosting performance.
- **Distributed Processing:** Improves resource efficiency by distributing queries to nodes where data resides. Inferences—such as scoring, relevance, and other ML model calculations—are performed locally, providing higher performance and avoiding network cost of moving data in centralized processing.

Addressing Latency and Reducing Cost (Continued)

- **Approximate Nearest Neighbor (ANN) Search:** Speeds retrieval by approximating nearest neighbors instead of exact matches, trading slight precision loss for significant gains in scalability and speed. ANN is ideal for large datasets where exact search is computationally costly, such as in real-time recommendations, interactive search, and time-sensitive applications like customer support or conversational AI.
- **Real-Time Indexing:** Processes and incorporates new or updated data instantly, allowing users to access the latest information without delay.
- **Rapid Indexing:** Employs techniques like quantization, which reduces each vector's value size (even down to a single bit per value) to accelerate indexing without reducing data points.
- **Embedding Compression:** Employs techniques like Matryoshka Representation Learning (MRL) and Binary Quantization Learning (BQL) to minimize storage requirements. MRL creates a flexible embedding hierarchy, while BQL compresses floating-point dimensions to 1-bit, greatly reducing storage needs with minimal accuracy loss.
- **Hierarchical Navigable Small World (HNSW) Indexing:** Offers tunable parameters (such as M for neighbor count and ef for exploration), enabling a balance between memory usage and accuracy. Some implementations support incremental graph updates, avoiding costly full re-indexing.
- **Late Interaction Models:** Precomputes document embeddings (for example, ColBERT) to reduce query-time computation, resulting in faster, more efficient searches.
- **Streaming Search**: Stores vectors on disk rather than in memory, significantly reducing storage costs while maintaining low-latency search, ideal for AI assistants handling large personal data volumes.

Security Considerations

As generative AI becomes more prevalent in the business, robust security is essential to protect sensitive data and maintain user privacy. RAG workflows often handle vast amounts of private information and span multiple systems, with significant security and governance implications.

The general approach taken by RAG vendors falls into two camps. RAG solutions from enterprise vendors such as IBM Watson Discovery incorporate data governance features, whereas solutions from platform vendors like Vespa.ai integrate with data governance frameworks to ensure compliance, data security, and data access and usage control.

Governance is greatly simplified with RAG architectures that minimize data movement. Where data remains in place during retrieval, ensuring governance policies remain intact.

Best Practices for Deploying RAG

For any organization, embracing generative AI can be daunting. The potential for business impact and transformation is vast. However, as with any new and complex technology, there is substantial potential for error and failure. To mitigate risk, Vespa advocates adopting an incremental crawl, walk, run approach where ambitions are aligned with experience and confidence.

Crawl: Experimentation

BM25 is an effective starting point for a RAG model. It uses keyword-based ranking to match query terms directly with document content, delivering strong relevance in text-heavy data without the need for embeddings. Unlike vector embeddings, BM25 requires no complex processing, as it operates on simple inverted indexes making setup straightforward and computationally light. This approach provides a strong baseline for RAG workflows, enabling generative models to create coherent responses based on contextually relevant text snippets retrieved via BM25.

Once you have a BM25-based RAG workflow in place, you can progressively move towards more advanced techniques by integrating vector embeddings when needed. Combining BM25 with vector embeddings can enhance retrieval quality further, as BM25 captures keyword relevance and embeddings capture semantic similarity. This hybrid approach outperforms either method on its own, but starting with BM25 lets you build a reliable foundation and scale up as requirements grow.

Walk: Enterprise Rollout

While proving the value of generative AI and gaining confidence with the required technology components is a vital first step, scaling AI across an entire enterprise brings unique challenges. These include seamless integration with existing data sources, stringent data privacy and security requirements, delivering high performance, and managing a complex, large-scale runtime environment. Scalability is a key concern, as AI models need to process massive and growing volumes of data while supporting diverse use cases—all without compromising on performance or reliability. Typically, these require more comprehensive platforms than those used for initial experiments.

Retrieval methods such as ColBert, which apply tensor computations in more advanced ways to achieve superior results will continue to proliferate as the work we and others are doing to make these economical at scale becomes more well known. Visual retrieval such as ColPali will continue to increase in popularity and gradually take over document search.

Run: Long Reasoning

Long reasoning, pioneered by OpenAI, is the current state of the art of generative AI. It describes a model's ability to enter into a multi-step reasoning dialog with a user to get to the heart of the matter – ultimately generating accurate and contextually relevant responses to intricate queries.

Each reasoning step may require information, which may translate into thousand of retrieval queries per problem, significantly impacting the RAG infrastructure. As a result, low latency becomes essential to maintain responsiveness, ensuring that end-users experience minimal delay even as the model engages in extensive reasoning. The infrastructure must cope with high request rates to handle this query surge efficiently and maintain system performance under heavy use. For internal RAG applications, such as those used within enterprises for knowledge management, research, or customer support, low-latency, high-throughput retrieval systems are paramount, as they enable long reasoning processes to operate effectively and deliver valuable insights quickly.

About Vespa.ai

Vespa.ai is a platform for building and running real-time AI-driven applications for search, recommendation, personalization, and RAG. It enables enterprise-wide AI deployment by efficiently managing data, inference, and logic, handling large data volumes and over 100K queries per second. Vespa supports precise hybrid search across vectors, text, and structured metadata. Available as both a managed service and open source, it's trusted by organizations like Spotify, Vinted, Wix, and Yahoo. The platform offers robust APIs, SDKs for integration, comprehensive monitoring metrics, and customizable features for optimized performance.

Interested to learn more? We have many different resources and information available through our social platforms

GitHub Twitter LinkedIn YouTube

Vespa.ai