RAG: Beyond the Chatbot

By Sam Charrington
Founder and Head of Research, TWIML
Host, TWIML AI Podcast

Introduction

ChatGPT's late 2022 launch commanded global attention, demonstrating the potential of combining large language models (LLMs) with a chat interface to access and interpret vast amounts of data.

Business leaders immediately asked, "How can I have a ChatGPT for my company's data?"

A popular answer to this question came in the form of *retrieval-augmented generation (RAG)*, a technique developed years prior by Facebook researchers. RAG extends language models by providing relevant contextual data from external sources, allowing more accurate, timely, and contextually grounded outputs than possible relying solely on the language model's embedded knowledge. This approach also helps address—but doesn't eliminate—the challenge of hallucination, where language models generate inaccurate information.

In recent months, many companies have begun building generative AI projects focused on delivering RAG-based chatbots for internal and external users, often starting with systems to augment traditional knowledge bases and search engines.

While these efforts are a great step at demonstrating the capability of generative AI, they often stumble due to the inherent limitations of chatbot interfaces.

This report aims to broaden technology leaders' perspectives on the opportunities that RAG offers, while informing their implementation strategies. It explores core concepts, real-world challenges, and high-value applications beyond simple chatbots.

Before discussing how retrieval-augmented generation works and helping you plan your deployment, let's first address the key limitations of chatbots in the enterprise. Doing so will allow us to better appreciate the broader approach advocated in this report.

SPONSORED BY
MongoDB.

Limitations of Chatbots

While ChatGPT's accessible chatbot interface was a key part of its appeal, dialogue-based interfaces have inherent weaknesses as enterprise tools. In particular, they often struggle to provide the structured and focused user experience that business users require for efficient task completion.

By integrating retrieval-augmented generation into established enterprise workflows—rather than delivering standalone chatbots—organizations can more quickly and effectively leverage its capabilities.

RAG-based systems integrated into enterprise workflows offer several key benefits:

1. **Focus:** Existing enterprise applications are designed to provide a focused user experience that guides users through complex tasks with precision. This is contrasted with open-ended chat interfaces that invite frustration due to lack of direction and mis-set expectations.
2. **Structure:** Many enterprise tasks revolve around handling discrete data values, such as financial figures, inventory numbers, customer details, or project milestones. Forcing users to interact with this structured data through a chat interface can be cumbersome and inefficient.
3. **Integration:** Integration with enterprise systems ensures that AI models have access to the latest relevant information for task completion, ensuring well-informed decisions and eliminating manual errors.
4. **Familiarity:** Employees are already trained on existing enterprise systems. Integrating RAG into familiar interfaces reduces the learning curve and increases adoption rates compared to introducing a new, standalone chatbot interface.
5. **Contextual Awareness:** Enterprise workflows often involve multi-step processes where each step provides context for the next. RAG integrated into these workflows can leverage and preserve this sequential context more effectively than a chatbot, which often lacks persistent state or process awareness.
6. **Security, Compliance, and Auditability:** Enterprise workflows often have built-in security measures, compliance checks, and auditing capabilities. Integrating RAG into these existing systems leverages established security protocols, ensuring data protection and regulatory compliance without having to reinvent these for standalone chatbots.

Comparing Standalone and Integrated RAG

Standalone Chatbot

Limitations
- Open-ended interface lacks focus
- Inefficient for structured data tasks
- Requires learning new interface
- Limited process awareness
- Separate security/compliance needs

Best for:
- Quick information lookup
- General Q&A
- Simple, standalone tasks

Integrated Workflow

Benefits
- Leverages familiar user interface
- Maintains existing security controls
- Preserves workflow context
- Structured for specific tasks
- Higher adoption rates

Best for:
- Complex business processes
- Multi-step workflows
- Enterprise-wide deployment

While chatbots have their place, they often fall short in complex enterprise environments. By integrating RAG into existing workflows and systems, organizations can take advantage of the technology's benefits while maintaining the structure, security, and familiarity that business users require. This approach paves the way for more effective and widely adopted AI solutions across the enterprise. Before exploring examples of how RAG can enhance a variety of business processes, let's review the key concepts underlying this technology.

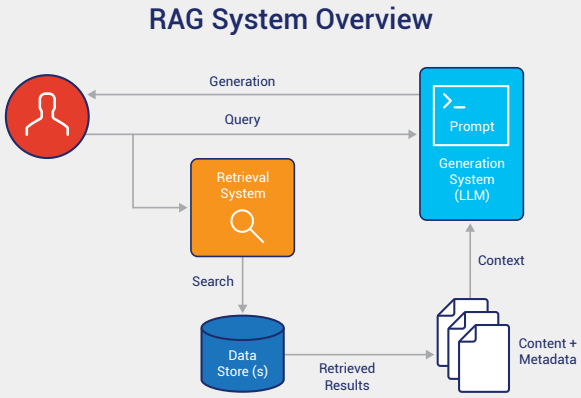
Core Concepts of RAG Systems

RAG systems consist of two main components, corresponding to the "R" and "G" in RAG:

1. **Retrieval system ("R")**
2. **Generation system ("G")**

The **retrieval system** responds to a user query by collecting the most precise and relevant context from the data store or grounding source. Key aspects include efficient data indexing and storage, and selecting and ranking the most relevant information.

The **generation system**, a large language model, takes the context returned by the retrieval system and, along with the user's prompt, uses it to generate a response to the user. It interprets and weaves the retrieved data into meaningful, coherent language.



The Importance of Balanced Focus

Early adopters often overemphasize RAG's "G" (generation) and neglect its "R" (retrieval). This imbalance can lead to a "garbage in, garbage out" scenario. No matter how sophisticated the LLM, if it's working with irrelevant or low-quality data, the responses will be subpar.

Retrieval is a complex problem that requires significant attention to ensure the language model gets high-quality, relevant data. Fortunately, we have decades of experience in information retrieval and search systems. It still requires substantial effort and discipline, particularly around evaluation metrics and systems, to deliver RAG-based products that users find genuinely helpful.

Retrieval Mechanisms

RAG doesn't specify a particular retrieval mechanism. Any method returning relevant context may be used. Popular approaches include:

1. **Vector search:** Converting text and other data types (images, audio files, etc.) into sequences of numbers, or "vectors," has been fundamental in computational sciences for over 50 years. Storing and querying these vectors in purpose-built databases grew in popularity in the 00s for semantic search applications and recommendation systems. Maturing just in time for use with LLMs, vector search systems have gained popularity in RAG implementations due to their ability to efficiently perform similarity search, which enables retrieving context that is relevant to the user's query based on meaning, rather than exact keyword matches.

2. **Traditional text-based search:** Despite the popularity of vector-based approaches, many modern RAG systems incorporate text- or keyword-based search to address the shortcomings of pure vector search, such as dealing with precisely worded queries, or rare or uncommon terms.

3. **Advanced approaches:** Emerging hybrid systems combine vector and text-based retrieval methods to leverage the strengths of each. Metadata-based filtering can be used to ensure only relevant context is selected. In two-stage systems a reranker provides a second stage of retrieval to identify the most relevant documents. Graph-based approaches use knowledge graphs to capture relationships between entities, enhancing contextual relevance.

Vector Embeddings: A Brief Explanation

If a vector represents text as a sequence of numbers, a vector *embedding* does so while preserving their meanings relative to one another.

A vector search system is one in which:

- Documents or text chunks are converted into high-dimensional vector embeddings.
- Queries are also converted to vectors.
- Similarity between query and document vectors is used to retrieve relevant information.

Canonical RAG System Pipeline

A typical retrieval-augmented generation system operates as follows:

1. **Data preparation:**
   - Collect the data for your context.
   - Chunk it into appropriately sized segments for your application.
   - Tag the chunks with metadata to aid in later retrieval.
   - Embed the chunks and store the embeddings in a vector database.
2. **Data retrieval**
   - Embed the user query.
   - Search the vector database for similar embeddings.
   - Retrieve the nearest-neighbor chunks.
3. **Response generation**
   - Add those chunks to a prompt for generating the response.
   - Perform an LLM inference or API call with this prompt.
   - Return the generated response to the user.

This workflow shows how the retrieval and generation components work together to provide contextually relevant responses based on your data.

By understanding these core concepts and the interplay between retrieval and generation, organizations can design and implement effective RAG systems for a variety of enterprise use cases.

RAG Beyond the Chatbot: Use Cases

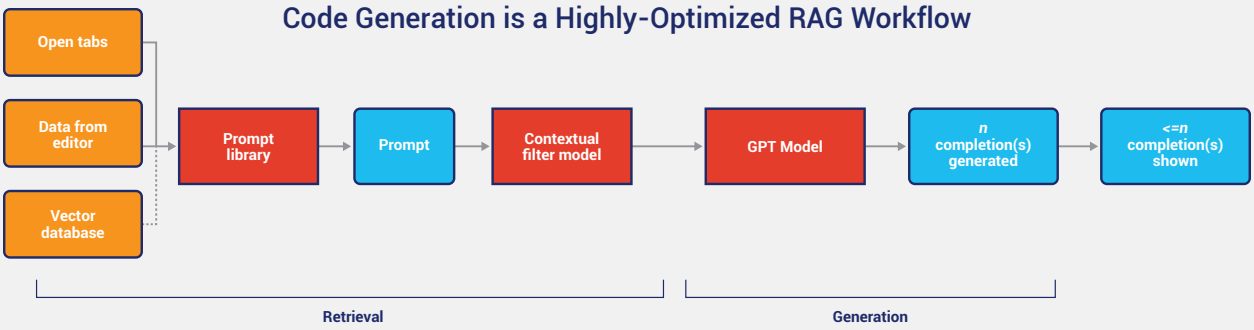
While chatbots are the prototypical RAG application, the technology's potential extends beyond conversational interfaces. RAG's ability to synthesize relevant business information opens up a wide array of enterprise applications.

This section explores diverse RAG use cases able to drive significant business value. By examining these examples, technology leaders can gain a broader perspective on applying RAG within their organizations to improve efficiency and drive innovation. Let's explore ways to deploy retrieval-augmented generation for business impact:

Code Generation

Though not typically considered a RAG application, code generation exemplifies RAG in action, where the retrieval context is the existing codebase and the generated content is code.

Code generation illustrates the importance of use-case-appropriate retrieval mechanisms. Unlike general RAG applications, code generation employs a combination of vector- and heuristics-based retrieval methods. For instance, GitHub Copilot utilizes an algorithm that emphasizes open tabs when prompting the language model for code completion.



Source: Github

This approach also demonstrates how RAG can be adapted to specific domains, leveraging contextual information (the developer's current focus) for more relevant suggestions. By prioritizing open tabs, the system can offer code completions that better align with the developer's task or thought process.

The evolution of code generation tools reinforces our thesis about workflow integration. While developers initially experimented with code generation through ChatGPT's interface, the technology's true potential emerged when tools like GitHub Copilot and Cursor embedded these capabilities directly into IDEs. This mirrors our broader argument: the greatest business value comes from integrating AI capabilities into existing tools and processes, not from standalone interfaces.

Document Generation

Content creation is a fundamental application of generative AI. By leveraging RAG, organizations can infuse core document generation processes with business-specific context, ensuring more accurate and relevant outputs.

Proposals: RAG can generate tailored proposals using a company's past successful proposals, pricing data, and project histories. Inputs can include relevant case studies, technical specifications, and client-specific information from internal file systems, CRM systems, and project management tools.

- **RFP Responses:** RAG can pull from a company's product documentation, customer support databases, and historical RFP responses, while incorporating up-to-date pricing and technical specifications. Data sources include file systems, product websites, support databases, and pricing systems.
- **Email:** RAG can support efficient 1:1 email communication by automatically crafting personalized, data-rich messages integrated with internal business systems and processes. It can draft tailored responses to sales and support inquiries using data from CRM systems, product documentation, and pricing and configuration management systems.
- **Legal:** RAG can streamline drafting legal documents and contracts. It can automatically generate legal drafts from company templates, clause libraries, and contract management systems, while ensuring compliance with the latest regulations and company policies.
- **Compliance:** RAG can support compliance processes by generating comprehensive documentation and reporting. It can create dynamic, up-to-date compliance manuals and reports from regulatory guidelines, industry standards, and training materials. It can produce detailed audit reports highlighting potential concerns based on templates and operational systems.

In these scenarios, RAG-based systems can provide relevant, accurate, and contextual information, reducing the time and effort for these complex tasks.

Dashboards & Decision Support

RAG can complement traditional dashboards by providing dynamic, context-aware insights:

- **Executive Summaries:** RAG can generate concise summaries of complex data from business intelligence tools, financial reports, and market analysis databases. Multimodal or vision-language models can interpret supplied graphs and charts.
- **Trend Analysis:** By accessing historical data and current market information, RAG can offer predictive insights based on financial and market research reports, helping decision-makers anticipate future trends.
- **Anomaly Detection:** RAG can highlight unusual data patterns from textual reports and log files based on historical norms and industry benchmarks.
- **Smart Alerts:** Tailoring alert content and delivery based on the recipient's role and preferences, using HR systems and user behavior data.

Research & Analysis

RAG-based systems can perform various research and analysis tasks:

- **Literature Reviews:** Automatically generating comprehensive literature reviews by analyzing academic databases, industry publications, and internal research documents.
- **Competitive Analysis:** Creating detailed reports on competitors by synthesizing information from news articles, financial reports, and social media data.
- **Patent Analysis:** Assisting patent research by analyzing patent databases, scientific literature, and internal R&D documents to identify innovation opportunities or potential infringements.

Education & Training

RAG integrated with learning management systems (LMSs), internal knowledge bases, and HR databases can support corporate learning and development goals:

- **Personalized Learning Journeys:** Create customized learning experiences and onboarding materials tailored to an employee's skills, career goals, role, and learning history.
- **Interactive Learning Materials:** Automatically produce tutorials, training modules, and assessments from user manuals, regulatory documents, and job-specific knowledge bases.
- **Skill Gap Analysis and Resource Matching:** Identify individual skill gaps by comparing job descriptions and performance reviews and suggesting relevant training resources from available learning catalogs and LMSs.

Field Support

RAG can support field operations organizations with applications such as:

- **Equipment Troubleshooting:** Generate step-by-step troubleshooting guides for field technicians, using equipment manuals, past incident reports, and expert knowledge bases.
- **Safety Protocols:** Provide context-aware safety information based on job site, environmental conditions, and equipment.
- **Customer Briefings:** Prepare field staff with relevant customer history and potential issues and opportunities before client visits.

Agentic Memory

LLMs have accelerated the development of agentic systems. These systems are designed to perform tasks, make decisions, and interact with tools and external systems to achieve objectives without continuous oversight. In agentic systems, retrieved context can serve as an agent's long-term memory, providing crucial information with which to conduct its activities.

The examples in this section demonstrate RAG's versatility, showcasing its ability to use diverse data sources to create intelligent, context-aware solutions across multiple domains.

Integrating RAG into Enterprise Workflows: A Roadmap

Setting up a basic RAG demo is relatively straightforward, but delivering a production-ready, user-focused, and tightly integrated solution that adds real business value requires careful planning. Here's a roadmap for implementing a robust, enterprise-grade RAG system:

1. Prioritizing Use Cases

- Identify pain points in existing processes where RAG can provide solutions.
- Assess potential ROI for key use cases.
- Consider workload reduction, process acceleration, and quality improvements.
- Evaluate potential impact on user satisfaction, team scalability, and competitive advantage.
- Balance short-term, high-impact initiatives with longer-term strategic investments.
- Begin with limited-scope pilot projects in non-critical workflows to build expertise and demonstrate value.

2. Analyzing Workflows

- Map existing workflows to pinpoint RAG integration opportunities.
- Identify key user interaction points for RAG enhancement.
- Determine where additional inputs might be needed to capture context for retrieval or generation.
- Explore how to present RAG-generated content alongside existing data.

3. Identifying Data Sources

- Map data flows in current business processes to identify RAG integration points.
- Identify relevant internal and external data sources in support of key workflows.
- Assess data quality, accessibility, and security.
- Explore necessary data preprocessing or cleansing steps.
- Consider data governance and compliance requirements.

4. Planning Evaluation

- Define clear metrics for measuring system performance and improvement to business outcomes.
- Build an evaluation dataset that reflects your real-world use cases.
- Ensure focus on retrieval relevance before approaching generation.
- Use your evaluation metrics to guide retrieval strategies and generative model selection.

5. Optimizing Retrieval

- Explore and improve retrieval strategies against established metrics.
- Identify metadata enrichment and data filtering opportunities.
- Choose suitable embedding models based on your use case and data types.
- Decide on chunking strategies for your documents.
- Evaluate database options.
- Consider performance and scalability needs.
- Explore advanced retrieval opportunities as needed

6. Refining Generation

- Start with the most capable models available when experimenting with generation pipelines.
- Assess various LLM options (open-source vs. proprietary, general vs. domain-specific) and parameters.
- Consider fine-tuning when appropriate for the use case.
- Review cost, capability, inference speed, and deployment implications.
- Confirm alignment with your organization's ethics and governance policies.

7. Building the System

- Develop an MVP for initial testing.
- Consider API- or plugin-based integrations that allow RAG to more easily interact with current systems.
- Implement robust error handling and logging.
- Ensure sufficient scalability and modularity for your needs.
- Follow MLOps/LLMOps best practices to streamline development and deployment.

8. Rolling Out & Gathering Feedback

- Plan a phased roll-out, starting with a pilot group and expanding gradually.
- Train users on how RAG enhances existing workflows.
- Implement continuous user feedback mechanisms.
- Use multiple rounds of A/B testing to compare different approaches.
- Establish a process for regular system audits and quality checks.
- Provide comprehensive documentation for RAG integration.
- Establish a support system for user queries and issues.
- Monitor system performance and user adoption closely.

9. Iterating and Improving

- Analyze user feedback and system performance data.
- Continuously update and improve your models and retrieval mechanisms.
- Improve RAG integration with existing systems over time, based on user feedback and changing needs.
- Review RAG technology advancements and incorporate them as appropriate.
- Reassess and adjust your use cases based on evolving business needs.

By following these steps and considering these key factors, organizations can implement RAG-based systems that deliver value and drive innovation across business functions.

RAG Roadmap

Case Study: scalestack

New York based Scalestack markets itself as an "AI-powered, enterprise-grade go-to-market (GTM) orchestration & activation platform." The Scalestack platform provides automated workflows that allow customer sales teams to clean, enrich and prioritize account data at scale, resulting in more complete buyer profiles, a greater ability to predict buying behavior, and a data-based way to prioritize sales and marketing efforts. Ultimately this reduces the amount of time sales reps spend researching prospects and increases their ability to focus on the right accounts at the right time.

Scalestack collaborated with MongoDB in March 2023 to develop Spotlight v1, an AI copilot for sales representatives. The initial user experience featured a chatGPT-inspired UI that displayed prioritized accounts with AI-generated insights. However, Scalestack soon discovered that sales representatives were reluctant to adopt another user interface, necessitating a crucial pivot in their approach.

Responding to user feedback, Scalestack shifted focus to integrating their AI capabilities into existing workflows and familiar tools. The result was a workflow builder that empowers users to create automated sequences for routine tasks without manual intervention. The insights generated by these RAG-based workflows can then be incorporated into the tools—like CRM systems—that its users and their managers already use.

Technical Implementation

Scalestack RAG implementation demonstrates sophisticated use of diverse data sources and advanced AI techniques. The company leverages MongoDB Atlas Vector Search for efficient data management and retrieval. This choice allows Scalestack to handle a wide range of data types with a flexible schema and powerful querying abilities over large datasets.

The company's RAG modules retrieve information from various sources, including:

- Customer-provided documents
- SEC reports
- Crawled websites
- API query results

Atlas Vector Search enables Scalestack RAG system to incorporate the most relevant company data from these sources, leading to dynamic, informative responses.

Scalestack implemented a comprehensive system for evaluation in order to maintain data quality and improve RAG performance:

1. **Tracing and Logging:** Each AI and machine learning call is traced and saved with input, output, and (for LLMs or agents) prompts and tools used.
2. **Continuous Improvement:** The company implemented a feedback loop that strongly prioritizes direct feedback from users. Any generated content that doesn't satisfy users is added to a test dataset and prompts and other aspects of the system are iterated on until the exact data that the client needs is produced.
3. **Testing Dataset:** A portion of queries, including traces receiving negative feedback, is automatically saved in a testing dataset. This dataset undergoes multiple checks, including:
    - AI checks (using an LLM as the evaluator)
    - Rule-based checks
    - RAG output checks
    - Human review

Testing and evaluation continue until the testing dataset passes all checks, aiming for a minimum of 99% accuracy in quality.

Impact

Scalestack's journey illustrates the power of integrating RAG directly into existing workflows. By pivoting from a standalone chatbot interface to an embedded solution, the company not only improved the value delivered to end users, but also enhanced the overall efficiency of their customers' sales processes. Their meticulous approach to data quality and continuous improvement demonstrates how RAG can be effectively implemented and refined in a real-world business context. This case study underscores a key theme of this report: the most impactful RAG implementations often augment familiar tools and processes to deliver tangible business value.

Conclusion

Retrieval-augmented generation opens up exciting possibilities for enterprises, far beyond simple chatbots. By integrating RAG with existing company workflows and data, organizations can create efficiencies and unlock new capabilities across a wide range of business functions. To fully capitalize on this potential, technology leaders must expand their perspective on RAG applications.

The key to success lies in identifying high-value use cases within your organization, then following a structured approach to implementation. Remember to focus first on the retrieval aspect of RAG; even the most advanced language models can't overcome the limitations of irrelevant or low-quality data.

As RAG technology evolves, we encourage you to look beyond chatbots and explore the myriad ways RAG can enhance your existing processes. By thoughtfully implementing RAG-enhanced workflows now, your organization can gain a competitive edge, driving efficiency and innovation across the enterprise.

About TWIML and Sam Charrington

Machine learning and artificial intelligence are dramatically changing how businesses operate and people live. Through our podcast, publications, and community, TWIML brings top minds and ideas from the world of AI to a broad and influential community of data scientists, engineers and tech-savvy business and IT leaders.

Sam Charrington is founder and head of research at TWIML, and host of The TWIML AI Podcast, the longest running and most popular podcast for ML and AI practitioners, researchers, and leaders, with over 700 episodes and 18 million downloads to date. Additionally, Sam is a sought after industry analyst, advisor, speaker, and thought leader. Sam's professional interests center on enterprise adoption of AI technologies, innovation storytelling and bringing AI-powered products to market, and AI-enabled and -enabling technology platforms.

About Our Sponsor

Thanks to our sponsor MongoDB for their support of this report.

Headquartered in New York, MongoDB's mission is to empower innovators to create, transform, and disrupt industries by unleashing the power of software and data. Built by developers, for developers, MongoDB is a database with an integrated set of related services that allow development teams to address the growing requirements for a wide variety of applications, all in a unified and consistent user experience. MongoDB has more than 50,000 customers in over 100 countries. The MongoDB database platform has been downloaded hundreds of millions of times since 2007, and there have been millions of builders trained through MongoDB University courses. To learn more, visit **mongodb.com**

Connect with us

sam@twimlai.com

@samcharrington

www.linkedin.com/in/samcharrington

@twimlai

twimlai.com

twiml