

Era of RAG-as-a-Service for Pretrained AI Applications

Byunghyun Ban^{1,2}

¹Department of Music, Gyeongbuk National University

²NaNa Lab

Andong-si, Republic of Korea

bhban@nanalab.kr

Abstract— Large language models are now used in many knowledge-intensive applications, but their outputs still need to be grounded in reliable, recent, and domain-specific information. Although many datasets and fine-tuning resources have been introduced for LLM development, many practical applications do not need to modify model weights; they need a convenient way to connect pretrained models to trusted external knowledge. Retrieval-Augmented Generation (RAG) has become a common approach for this purpose, but production RAG requires document ingestion, chunking, embedding, indexing, retrieval, reranking, access control, citation handling, and continuous evaluation. This short paper discusses RAG-as-a-Service (RAGaaS) as an emerging service model that provides retrieval, grounding, curation, and citation as a managed knowledge delivery layer. Unlike vector databases or enterprise search alone, RAGaaS is defined here as a reusable service interface that connects pretrained models to curated and citation-ready knowledge. We distinguish RAGaaS from adjacent concepts, review representative services and downstream implementations, classify likely demand areas, and propose practical criteria for evaluating RAGaaS platforms.

Keywords— *Retrieval-Augmented Generation, RAG-as-a-Service, Knowledge-as-a-Service, Knowledge Delivery, Citation, Enterprise AI*

I. INTRODUCTION

Recent AI systems are being applied across academic, industrial, legal, medical, educational, and administrative domains. As LLMs are increasingly used in scholarly and professional workflows, their responsible use and precise grounding are increasingly required [1]. The issue is not only whether a model can generate fluent text, but whether its output can be checked, traced, and trusted.

At the same time, data resources for LLM development are growing quickly. Language-specific datasets, general language understanding benchmarks, multitask evaluation sets, instruction-tuning corpora, and alignment datasets have all been introduced to improve model training and evaluation [2-8]. These efforts show that data remains one of the most important resources in the LLM ecosystem.

However, not every AI application needs additional training or fine-tuning. In many cases, the pretrained model is already strong enough as a language interface. The harder problem is giving that model access to current, curated, domain-specific, and citable knowledge. A customer-support assistant needs the latest product manual. A legal assistant needs the relevant clause or statute. A research assistant needs a source that can actually be cited. For these cases, changing the model weights may be less practical than connecting the model to an external knowledge source.

RAG has attracted attention because it provides a practical way to connect pretrained models with external documents at inference time [9-16]. The idea is straightforward: retrieve relevant knowledge, insert it into the model context, and generate an answer based on that evidence. In production, however, the work is not so simple. A RAG system often requires document ingestion, parsing, chunking, embedding, indexing, hybrid retrieval, reranking, prompt construction, citation management, permission control, and evaluation [17-20].

For small teams and domain users, the practical question is often not how to train a better model, but how to connect an already capable pretrained model to trustworthy knowledge. This paper argues that RAG-as-a-Service (RAGaaS) answers this gap by shifting the burden of retrieval and citation infrastructure from the application developer to a managed service provider.

II. WHAT MAKES RAGaaS DIFFERENT?

The term RAGaaS can easily become vague. If every vector database, search engine, or chatbot with documents is called RAGaaS, the term loses value. Therefore, this paper uses RAGaaS in a narrower sense.

In this paper, RAGaaS refers to a managed service layer that provides retrieval, corpus management, grounding, citation, and governance functions for pretrained AI applications through reusable interfaces.

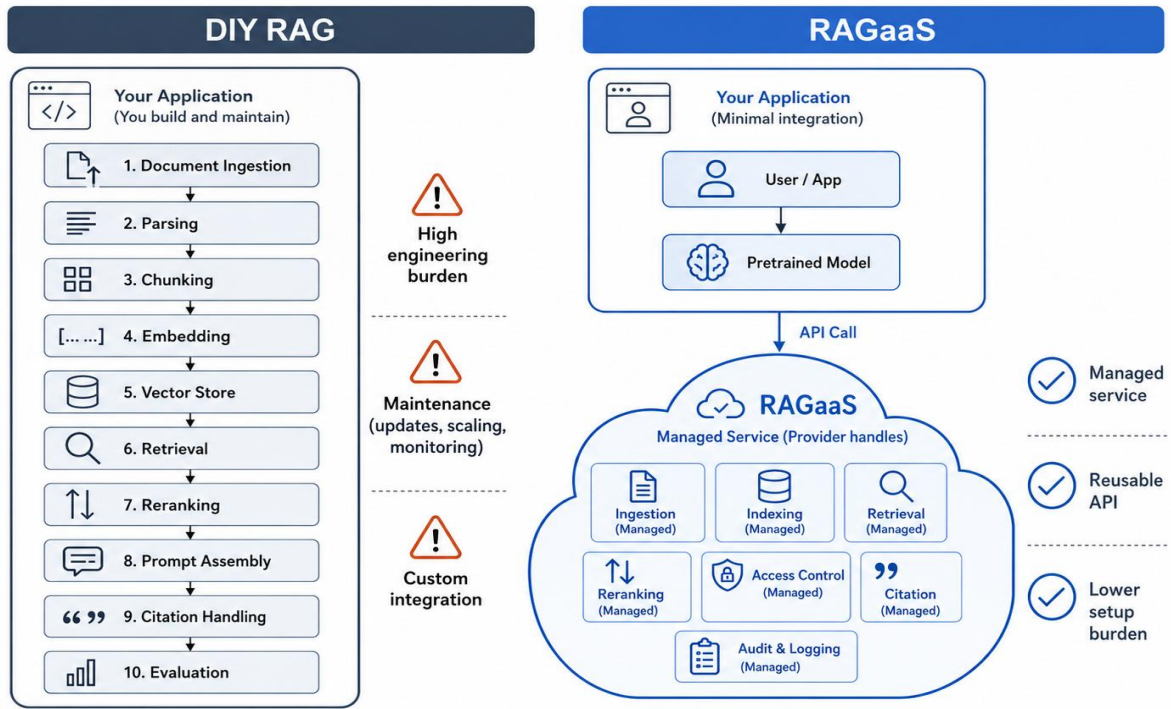


Fig. 1. DIY RAG vs RAG-as-a-Service. Comparison between building a full RAG pipeline inside an application and using RAGaaS as an external managed knowledge delivery layer.

TABLE I. RAGAAS AND ADJACENT CONCEPTS

Concept	Main function	Limitation when used alone	Relation to RAGaaS
Vector database	Stores and retrieves embeddings	Usually does not handle full ingestion, citation, access control, or answer grounding	Can be a component of RAGaaS
Enterprise search	Searches documents across an organization	Often optimized for human search, not generative answer grounding	Can provide retrieval input to RAGaaS
Fine-tuning	Modifies model behavior or style	Inefficient for fast-changing facts and does not automatically provide citations	Alternative to retrieval-based knowledge delivery
Long-context prompting	Puts many documents directly into context	Costly and hard to scale for large or dynamic corpora	Useful for small-scale cases
DIY RAG	Full in-house RAG pipeline	High engineering and maintenance cost	What RAGaaS tries to simplify
Managed RAG	Managed implementation of a RAG pipeline	May focus mainly on retrieval infrastructure	Early or partial form of RAGaaS
RAG-as-a-Service	Managed retrieval, curation, citation, access control, and evaluation	Requires trust in provider and clear governance	Knowledge delivery layer for pretrained AI applications

This definition is intentionally broader than a vector database, but narrower than the entire AI application. RAGaaS does not necessarily replace the LLM. It also does not necessarily own the end-user interface. Instead, it sits between the pretrained model and the knowledge corpus.

The key difference is service boundary. In a conventional RAG system, retrieval and citation are internal modules of one

application. In RAGaaS, they become reusable service interfaces. A single knowledge layer may support multiple models, products, departments, or agents.

The term citation-ready is also important. In this paper, citation-ready means that retrieved evidence is returned with enough metadata for verification, such as document title, source identifier, section or page, retrieval timestamp, access status, and

the passage used to support the generated claim. A citation-ready system does not merely retrieve text. It returns evidence in a form that can be inspected.

III. RAGaaS ECOSYSTEMS

Several current services already show the direction of RAGaaS, although they use different names.

Amazon Bedrock Knowledge Bases supports a managed RAG workflow for connecting foundation models and agents to enterprise data sources [21]. Google Vertex AI RAG Engine presents RAG as a managed service for building context-augmented LLM applications [22]. Microsoft Azure AI Search has introduced agentic retrieval, where an LLM helps decompose complex questions into subqueries for RAG and agent workflows [23].

Other services are closer to an application-facing knowledge layer. Pinecone Assistant describes a managed knowledge layer that handles document processing, embeddings, retrieval, query planning, reranking, and answer generation [24]. Vectara positions its platform around managed RAG for enterprise assistants and agents [25]. Meilisearch explicitly discusses RAG-as-a-Service as a service model involving retrieval, hybrid search, citations, and access control [26]. Cohere Rerank does not provide a full RAGaaS platform, but it supports the ecosystem by improving the quality of retrieved passages before they are passed into the model [27].

Rather than treating these products as identical, it is more useful to classify them by function. This classification avoids treating vendor products as the main contribution of the paper. The products are evidence of a broader shift: retrieval, grounding, and citation are moving from local implementation details to service-level infrastructure.

IV. PRODUCTS AND IMPLEMENTATIONS BUILT ON MANAGED RETRIEVAL SERVICES

RAGaaS is not only a provider-side infrastructure category. It is also becoming a foundation for downstream AI products and enterprise workflows. The clearest cases are not always labeled "RAGaaS" by vendors, but they share a common pattern: application builders separate the user-facing AI product from the managed retrieval and knowledge-delivery layer.

CustomGPT.ai is a useful downstream SaaS example. Pinecone describes CustomGPT.ai as a no-code platform that helps businesses build domain-specific GPTs connected to their own data for use cases such as employee training, helpdesk automation, content generation, and knowledge management [29]. This case shows how a user-facing AI product can be built on retrieval infrastructure while focusing on workflows, integrations, attribution, and user experience.

Ring provides a stronger managed RAG example. AWS reports that Ring built a production-ready, multi-locale RAG-based support chatbot using Amazon Bedrock Knowledge Bases [30]. The system uses metadata-driven filtering for region-specific content and separates ingestion, evaluation, and

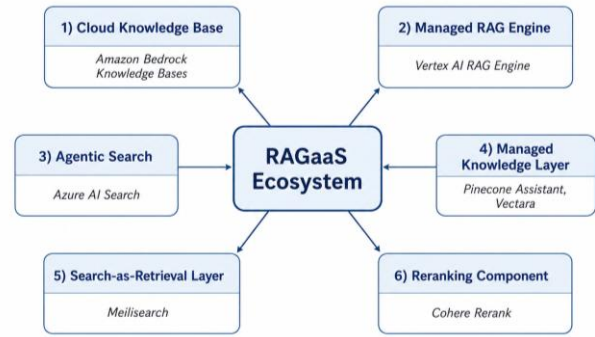


Fig. 2. Functional taxonomy of the RAGaaS ecosystem. Representative service types in the RAGaaS ecosystem, including cloud knowledge bases, managed RAG engines, agentic search, managed knowledge layers, search-as-retrieval services, and reranking components.

promotion workflows. AWS also reports that Ring reduced the cost of scaling to each additional locale by 21% while maintaining consistent customer experiences across 10 international regions [30].

Lotte Shopping is another strong enterprise workflow case. AWS Korea describes a Slack-based AI operation assistant that uses Amazon Bedrock Knowledge Bases to retrieve operational documents from Confluence, Wiki, AWS documentation, and accumulated operational knowledge [31]. The system also integrates Amazon Bedrock Agents, Slack, JIRA, and RCA report generation, showing how managed retrieval can become part of an operational response workflow.

Cox Automotive shows how managed retrieval can support agentic systems. AWS states that Amazon Bedrock Knowledge Bases handles RAG pipelines for Cox Automotive, searching thousands of documents in milliseconds using Amazon Titan embeddings [32]. In this architecture, the retrieval layer supports AI agents together with Amazon Bedrock, AgentCore, guardrails, and observability tools.

Other cases should be treated more cautiously. SonoSim's Vectara-based question-answering system is a useful domain-specific QA example, but public materials do not necessarily label it as RAGaaS [33]. Kiwa Digital's Caitlyn and CultureQ are relevant as Bedrock-based generative AI applications for cultural knowledge preservation, but the available public AWS materials do not clearly identify Amazon Bedrock Knowledge Bases as the retrieval layer [34]. Conversica's Vectara case is better treated as an AI-search predecessor rather than a clean RAGaaS implementation [35].

These cases are not identical, and not all of them call themselves RAGaaS. Still, they show the same architectural movement. Application builders are increasingly separating the user-facing AI product from the knowledge delivery infrastructure. The product may be a chatbot, assistant, agent, or no-code platform. Underneath it, a managed retrieval layer provides the current and domain-specific knowledge that the pretrained model needs.

This matters for the argument of the paper. RAGaaS is not only an abstract category. It is already visible in products and

workflows where the value comes from connecting pretrained models to curated corpora without building the entire RAG stack internally.

V. IMPORTANCE OF KNOWLEDGE DELIVERY

The demand for RAGaaS comes from a simple observation. Many AI applications do not mainly need a new model. They need better access to knowledge.

In enterprise settings, useful knowledge is scattered across internal documents, wikis, PDFs, ticket systems, customer histories, product databases, and shared drives. Employees may ask questions that require current policy documents, technical manuals, project records, or customer-specific information. A general pretrained model cannot be expected to know these materials. A RAGaaS layer can make them searchable and usable without requiring every organization to build a retrieval stack from the beginning [20-23].

Customer support is another natural case. Support assistants must answer based on product manuals, warranty conditions, troubleshooting guides, release notes, and known issues. These documents change frequently. A model-only system may generate a fluent but outdated answer, while a RAG-based system can retrieve the current source and include supporting evidence [21,24-26,30].

Citation-sensitive work is especially important. Legal, policy, compliance, and research tasks require more than plausible answers. They require references. In these fields, a generated answer is useful only if the user can check the supporting passage. This is where RAGaaS can become more than a search tool. If the service returns document title, section, date, page, URL, permission status, and confidence score, it becomes a citation layer.

This point also applies to research and literature review. AI tools are increasingly used to summarize papers, compare claims, and assist scholarly workflows. Yet academic work depends on attribution and verifiability. If a model invents a citation, misreads a source, or cites a paper that does not support its claim, the issue is not just technical. It becomes a research-integrity problem [1]. A RAGaaS system designed for scholarly corpora could provide fast retrieval over curated papers and return citation-ready evidence.

These use cases can be organized not only by domain, but also by the type of knowledge demand. This classification also explains why RAGaaS may grow even as pretrained models improve. A better model does not remove the need for fresh, permissioned, and verifiable knowledge. In many settings, the model is not the scarce resource. The scarce resource is reliable knowledge.

VI. REFERENCE ARCHITECTURE AND EVALUATION CRITERIA

A RAGaaS-based application can be described with a simple workflow:

User Query

- > Application
- > Pretrained Model
- > RAGaaS API
- > Curated Corpus
- > Citation-Aware Answer
- > Evaluation and Audit Log

The corpus may include company documents, manuals, research papers, laws, reports, tickets, product pages, and structured records. The important point is not only that data exists, but that it is curated. Old, duplicated, contradictory, or low-quality documents can easily damage the output of a RAG system. The RAGaaS API handles the retrieval side. It may parse documents, chunk them, create embeddings, build indexes, apply hybrid search, rerank candidates, and return relevant passages. In a mature service, the retrieved result should include metadata such as source, section, timestamp, relevance score, and access status. The citation layer is what makes the service useful in professional contexts. A retrieved passage should be returned as evidence, not just as hidden context for the model. This means that the answer can show where a claim came from and whether the cited passage actually supports it. The pretrained model then becomes only one part of the application. The model may be replaced or upgraded, while the curated knowledge layer remains. This is one reason RAGaaS is attractive: it separates model access from knowledge delivery.

To compare RAGaaS platforms, evaluation criteria are needed. RAG evaluation is already an active topic, with work on faithfulness, retrieval quality, factuality, and system-level performance [18,19]. For RAGaaS, these criteria should be extended to include service-level concerns.

VII. CHALLENGES AND FUTURE DIRECTIONS

RAGaaS is useful because it hides complexity. But hiding complexity does not remove it.

The first challenge is privacy and access control. A retrieval system may touch sensitive documents, customer data, contracts, internal policies, or personal records. If access control is weak, the model may reveal information that the user should not see. This is especially serious when RAGaaS connects many data sources into one knowledge layer [23,28].

The second challenge is citation faithfulness. A model may produce a citation that looks correct but does not actually support the sentence it is attached to. For RAGaaS to be trusted in academic, legal, medical, or compliance settings, it must evaluate whether cited evidence really supports the generated answer [18,19].

The third challenge is latency and cost. RAG systems often combine retrieval, reranking, prompt augmentation, and generation. Each step can improve quality, but each step may

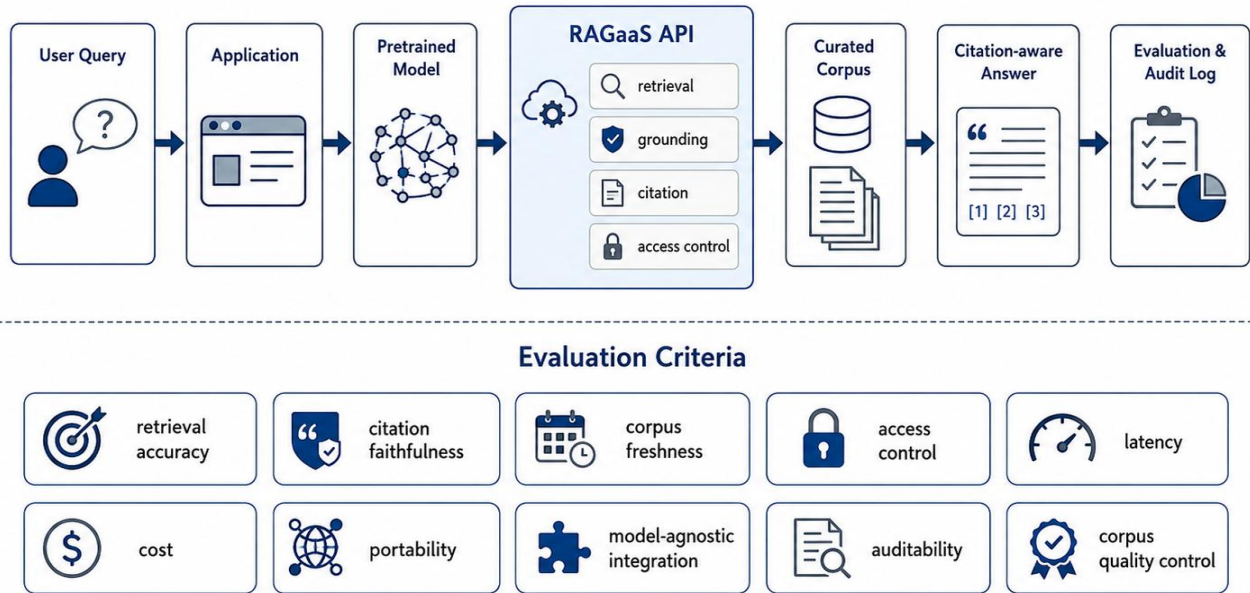


Fig. 3. Reference architecture and evaluation criteria for RAGaaS. A RAGaaS-based application connects a pretrained model to a curated corpus through retrieval, grounding, citation, and access-control APIs.

also increase response time and cost. Reranking can reduce irrelevant passages, but it adds another processing stage [27].

The fourth challenge is vendor lock-in. If embeddings, indexes, metadata schemas, and citation formats are proprietary, users may find it difficult to move from one provider to another. This issue may become more important as organizations build multiple applications on top of managed knowledge layers.

The fifth challenge is corpus quality. RAG does not automatically make an application reliable. If the knowledge base is stale, noisy, biased, or contradictory, the system may still generate bad answers. In the long run, successful RAGaaS platforms may need to provide not only retrieval APIs but also tools for corpus cleaning, deduplication, authority ranking, freshness monitoring, and conflict detection.

A final challenge is evaluation at the service level. Many current RAG evaluations focus on the answer quality of a single application. RAGaaS requires a broader view. A service may support many applications, models, and corpora. Therefore, future work should ask how managed knowledge layers can be evaluated across retrieval quality, citation faithfulness, privacy, portability, latency, localization, and corpus governance.

VIII. CONCLUSION

RAG has become a common method for connecting LLMs to external knowledge, but production RAG is not easy to build or maintain. RAG-as-a-Service responds to this problem by turning retrieval, grounding, curation, and citation into a managed service layer.

The main argument of this paper is that RAGaaS should not be understood merely as managed vector search. It is better

viewed as a knowledge delivery layer for pretrained AI applications. This layer allows application builders to connect capable pretrained models to curated, permission-aware, and citation-ready knowledge without operating the entire RAG pipeline themselves.

The paper distinguished RAGaaS from adjacent concepts, organized representative services into a functional taxonomy, reviewed downstream products and implementations, and proposed evaluation criteria for RAGaaS platforms. The implementation cases are important because they show that RAGaaS is not only a provider-side infrastructure category. It is already being used under customer-support chatbots, no-code GPT products, operational assistants, domain QA systems, and agentic enterprise workflows.

As pretrained models become easier to access, many applications may compete less on model ownership and more on the quality of the knowledge they can retrieve, verify, and cite. In that sense, the next stage of AI application development may depend as much on knowledge infrastructure as on model capability.

REFERENCES

- [1] B. Ban, "Can LLMs Review Scientific Papers?" ResearchGate preprint, 2026, doi:10.13140/RG.2.2.34160.83209.
- [2] B. Ban, "A Survey on Awesome Korean NLP Datasets," in 2022 13th International Conference on Information and Communication Technology Convergence (ICTC), pp. 1615-1620, 2022; arXiv:2112.01624.
- [3] A. Wang et al., "SuperGLUE: A Stickier Benchmark for General-Purpose Language Understanding Systems," Advances in Neural Information Processing Systems, 2019.

- [4] D. Hendrycks et al., "Measuring Massive Multitask Language Understanding," International Conference on Learning Representations, 2021.
- [5] S. Longpre et al., "The Flan Collection: Designing Data and Methods for Effective Instruction Tuning," International Conference on Machine Learning, 2023.
- [6] R. Taori et al., "Stanford Alpaca: An Instruction-following LLaMA Model," Stanford CRFM, 2023.
- [7] A. Koepf et al., "OpenAssistant Conversations: Democratizing Large Language Model Alignment," arXiv:2304.07327, 2023.
- [8] M. Conover et al., "Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM," Databricks, 2023.
- [9] P. Lewis et al., "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks," Advances in Neural Information Processing Systems, 2020.
- [10] K. Guu et al., "REALM: Retrieval-Augmented Language Model Pre-Training," International Conference on Machine Learning, 2020.
- [11] V. Karpukhin et al., "Dense Passage Retrieval for Open-Domain Question Answering," Empirical Methods in Natural Language Processing, 2020.
- [12] G. Izacard and E. Grave, "Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering," arXiv:2007.01282, 2020.
- [13] S. Borgeaud et al., "Improving Language Models by Retrieving from Trillions of Tokens," International Conference on Machine Learning, 2022.
- [14] A. Asai et al., "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection," International Conference on Learning Representations, 2024.
- [15] Y. Gao et al., "Retrieval-Augmented Generation for Large Language Models: A Survey," arXiv:2312.10997, 2023.
- [16] S. Zhao et al., "Retrieval Augmented Generation and Beyond: A Comprehensive Survey on How to Make Language Models Use External Data More Wisely," arXiv:2409.14924, 2024.
- [17] D. Edge et al., "From Local to Global: A Graph RAG Approach to Query-Focused Summarization," arXiv:2404.16130, 2024.
- [18] S. Es et al., "RAGAs: Automated Evaluation of Retrieval Augmented Generation," Proceedings of EACL System Demonstrations, 2024.
- [19] A. Gan et al., "Retrieval Augmented Generation Evaluation in the Era of Large Language Models: A Comprehensive Survey," arXiv:2504.14891, 2025.
- [20] E. Karakurt and A. Akbulut, "Retrieval-Augmented Generation (RAG) and Large Language Models (LLMs) for Enterprise Knowledge Management and Document Automation: A Systematic Literature Review," Applied Sciences, vol. 16, no. 1, article 368, 2026.
- [21] Amazon Web Services, "Knowledge Bases for Amazon Bedrock," AWS Documentation/Product Page, accessed May 2026.
- [22] Google Cloud, "Vertex AI RAG Engine," Google Cloud Documentation/Product Blog, accessed May 2026.
- [23] Microsoft, "Agentic Retrieval in Azure AI Search," Microsoft Learn, accessed May 2026.
- [24] Pinecone, "A Managed Knowledge Layer for Production AI Applications," Pinecone Blog, accessed May 2026.
- [25] Vectara, "Retrieval-Augmented Generation: Everything You Need to Know," Vectara Product Resource, accessed May 2026.
- [26] Meilisearch, "RAG-as-a-Service: What It Is, Use Cases, Providers and More," Meilisearch Blog, accessed May 2026.
- [27] Cohere, "Rerank: Boost Enterprise Search and Retrieval," Cohere Product Documentation, accessed May 2026.
- [28] European Data Protection Supervisor, "Retrieval-Augmented Generation," EDPS TechSonar, accessed May 2026.
- [29] Pinecone, "Domain-specific AI agents at scale: CustomGPT.ai serves 10,000+ businesses with Pinecone," Pinecone Customer Story, accessed May 2026.
- [30] Amazon Web Services, "How Ring scales global customer support with Amazon Bedrock Knowledge Bases," AWS Machine Learning Blog, 2026.
- [31] Amazon Web Services, "롯데쇼핑의 Amazon Bedrock 기반 AI 운영 어시스턴트 구축 사례," AWS Korea Tech Blog, 2025.
- [32] Amazon Web Services, "How Cox Automotive launched AI agents at scale using Amazon Bedrock AgentCore," AWS Customer Case Study, accessed May 2026.
- [33] SonoSim, "Vectara and SonoSim Announce Partnership to Bring LLMs to Healthcare Training," SonoSim News, 2023.
- [34] Amazon Web Services, "Revitalizing Indigenous Language with AI: Kiwa Digital + Caitlyn on AWS," AWS Partner Network Blog, 2025.
- [35] Vectara, "Conversica Case Study," Vectara Customer Story, accessed May 2026.