Journal of Systemics, Cybernetics and Informatics (2025) 23(6), 27-35
https://doi.org/10.54808/JSCI.23.06.27

Case Study on Understanding the Power of Retrieval Augmented Generation (RAG)

Venkata Jaipal Reddy BATTHULA
Department of Computer Science, Southwestern College
Winfield, KS 67156 USA

Richard S. SEGALL
Department of Information Systems & Business Analytics, Arkansas State University
Neil Griffin College of Business
State University, AR 72467 USA

Sreejith SIVASUBRAMONY
Sr. Data Scientist, Walmart Inc.
Bentonville, AR 72716 USA

ABSTRACT

This paper explores how Generative AI is changing with the use of Retrieval-Augmented Generation (RAG). RAG helps improve Artificial Intelligence (AI) systems by making them more capable, efficient and accurate. The paper explains how to build the Retrieval Augmented Generation model, covering important steps like preparing the data, creating embeddings, and setting up the retrieval system. Through a case study, we look at the main components of RAG, how it works with Large Language Models (LLMs), and why it is important in everyday digital tools. One of the goals is to compare different strategies for RAG, including choices for embeddings, similarity metrics and language models to find an optimal approach that can be generalized to work best. This helps us to understand how these factors affect performance and gives us ideas for building better and more efficient systems.

Keywords: Artificial Intelligence (AI), Large Language Models (LLMs), Natural Language Processing (NLP), Retrieval-Augmented Generation (RAG).

1. INTRODUCTION

According to Rothman (2025), Retrieval Augmented Generation (RAG) is a framework that was created because advanced generative Artificial Intelligence (AI) models can only generate responses based on the data they have been trained on. RAG retrieves relevant data from external sources in real-time and uses this data to generate more accurate and "contextually relevant responses". Rothman (2025) claims that one of the key strengths of RAG is its "adaptability" in that it can be applied to any type of data: text, images or audio.

One of the most cited papers on RAG is that of Goyal et al. (2020) who introduced RAG models where the parametric memory is a pre-trained sequence-to-sequence model and the non-parametric memory as a dense vector index of Wikipedia, accessed with a pre-trained neural received.

Other most cited works on RAG include that by (1.) Siriwardhana et al. (2023) who improved the domain adaption of retrieval augmented generation (RAG) models for open domain question answering., and (3.) Chen et al. (2024) who discussed benchmarking Large Language Models (LLMs) in RAG.

According to Bourne (2025), RAG has been shown to be useful for companies across many industry-types to improve their products, services and operational efficiencies that include customer support and chatbots, technical support, and automated reporting, and automated reporting that uses unstructured data and converts into more comprehensible formats.

The three stages of RAG are (1) Indexing, (2) Retrieval, and (3) Generation such as shown in Figure 1. Many recent works have been written in 2023 through 2025 on mastering RAG models for enhancing Natural Language Processing (NLP) and improved text generation of Large Language Models (LLM) that include those such as by Allen (2024), Baker (2025), Bhoyar (2024), Bourne (2025), Brener (2025), Johnson (2025), Lisztin (2025), Rothman (2025), Smith (2025), and Taylor (2025).

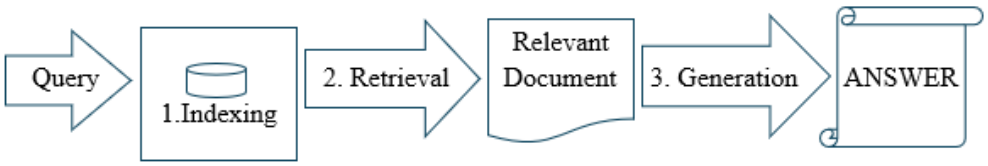

Figure 1: Framework of 3 Stages for Retrieval Augmented Generation (RAG)
(Acknowledge constructing and deriving this Figure 1 from similar figures in Chapter 4 of Bourne (2025).)

Do Rosario (2023) claims 2024 as the "Year of RAG" seeing significant adoption of this technology. De Rosario (2023) provides a simple definition of RAG as "an AI framework for retrieving facts from an external knowledge base to ground Large Language Models (LLMs) on the most accurate, up-to-date information and to give users insight into LLMs' generative process". The first half of 2024 saw significant progress in RAG applications like question answering and customer service, largely due to the convergence of open-source and commercial LLMs. Figure 2 in the Appendix titled "2024: Year of the RAG" provides a more detailed figure of RAG that also includes citation calculation algorithm, embedding generation services, anti-hallucination algorithm, and chunk ranking algorithm.

Graph-based approach to Retrieval Augmented Generation (RAG) has been investigated by Sprinter (2025) and Devline (2025) to integrate knowledge Graphs with Large Language Models (LLMs) to create Graph RAG LLM as discussed in their separate books devoted to this topic. Ability to incorporate graph structures in order to facilitate reasoning (Ding et al., 2005) and to generate graphs that express knowledge (Dickerson et al., 2003) are important in biological as well as other domains, motivating further progress in this direction.

1.1 Traditional NLP and its limitations

Natural Language Processing (NLP) has helped computers reason with human language. Prior to LLMs, most NLP relied on probabilistic reasoning as rule-based approaches. For example, "If we see the word 'Mr.' or 'Mrs.', it could be a person's name." These rule-based systems will work for simple tasks but might fail when things get complex.

Later, machine learning and deep learning research became popular, and models started learning patterns from data rather than just relying on hand-written rules. NLP techniques improved and computers became better at tasks like language translation and question answering.

But even advanced NLP models have a problem: they can only answer questions based on what they were trained on. If they were queried on something new, they will not know the answer and might also try to make up an answer that isn't right. This is a problem, especially when people are using the software to get important information.

There is an increasing demand for NLP systems that can perform well in settings where the scope of queries is vast and constantly evolving. LLMs are trained on huge volumes of data and use billions of parameters to generate original output for tasks like completing sentences and question answering (Allen, 2024). Despite the impressive performance of standalone LLMs, they are limited by the data they are trained on, which might be outdated. It is not feasible to constantly retrain and scale the models with the latest data because of the computational and environmental costs. (Nucci, 2024).

Retrieval-Augmented Generation (RAG) extends the already powerful capabilities of LLMs to specific domains or an organization's internal knowledge base, all without the need to retrain the model. It is a cost-effective approach to improving LLM output so it remains relevant, accurate, and useful in various contexts (Amazon, 2024). RAG works by optimizing the output of a Large Language Model (LLM) by letting the model reference a knowledge base outside of its training data before generating a response. Thus, the model does not rely on just its memory and is able to get the latest facts before answering a question. It combines the best of two worlds: information retrieval and generative modeling.

In simple terms, RAG retrieves relevant documents from a large corpus using a retrieval mechanism, and then uses a generative model, which is transformer-based, to convert the retrieved information to relevant useful output. The retrieval component ensures that the model has access to the latest data and task-specific knowledge, while the generation component allows for relevant answer creation. This dynamic retrieval-generation approach helps bridge

the gap between static training data and real-world information. RAG is scalable and also reduces the risk of hallucinations since it dynamically extracts relevant text at inference time (Superannotate, 2025).

2. CASE STUDY

This case study explores the role of chunking in RAG pipelines, the importance of embedding generation, the function of vector databases, and the impact of different similarity metrics on retrieval performance. It also compares RAG with alternative architecture.

2.1 Understanding Chunking in RAG

In Retrieval-Augmented Generation (RAG), chunking is the process of breaking down large texts into smaller, more manageable portions known as chunks as shown as Step C in Figure 3 in the Appendix. These can range in length from complete paragraphs to single sentences and are limited by token count. This text structure improves retrieval, which is crucial to the overall efficiency of RAG systems (Bourne, 2025). Chunking enables the model to match user queries to the most relevant parts of the source text. It maintains context for both individual and generic requests while also speeding up the response generation process.

Efficient retrieval is an essential feature in RAG systems. When working with large corpora (often millions of words), chunking becomes a crucial step. It ensures that the model can quickly identify the most relevant bits of text, which streamlines the answer generating process. For chunking, we experimented with:

Fixed-length chunks: Dividing the text into equally sized sections.

Sentence-based chunks: Splitting documents based on sentence boundaries for better context preservation.

Topic-based chunks: Grouping related paragraphs together to ensure context remains relevant.

2.2 Embedding Generation

After chunking, each segment of text is transformed into a *vector embedding* — a high-dimensional numerical representation (e.g. vector) that captures the semantic meaning of the content as shown in Step D in Figure 3 in the Appendix. This transformation is typically done using language models such as BERT, OpenAI's embedding models, or other transformer-based architectures. Text chunks are transformed into high-dimensional vectors, often with hundreds of dimensions, where each dimension captures part of the text's meaning. In this vector space, semantically similar content clusters together, enabling the system to retrieve relevant information based on meaning rather than exact wording. To maintain consistency and ensure accurate comparisons, the same embedding model is used for both the document chunks and user queries**.**

In our experiment, we tested different embedding models (OpenAI text-embedding-ada-002, sentence transformer MiniLM and bge-small) to convert text into numerical representations that the model can understand.

2.3 Vector Databases

Vector databases are purpose-built systems designed to efficiently store and retrieve high-dimensional vector embeddings. Tools like Qdrant, Weaviate, and others have become foundational components in Retrieval-Augmented Generation (RAG) pipelines, as they enable fast and accurate semantic searches across large datasets.

These databases are engineered for scalability, capable of handling vast collections of vectorized data—often reaching into the billions. This makes them well-suited for high-volume, enterprise-level applications. To maintain retrieval speed and accuracy, they leverage advanced indexing algorithms such as Hierarchical Navigable Small World (HNSW), which significantly accelerate the process of finding the most relevant vectors based on similarity.

In addition to handling dense embeddings that capture semantic meaning, many vector databases also support hybrid search functionality. This allows them to process both dense vectors and sparse, keyword-based vectors simultaneously, enabling a more flexible and comprehensive approach to information retrieval. Such versatility makes vector databases a powerful tool for building intelligent, context-aware AI systems.

2.4 Information Retrieval

When a user submits a query in a RAG system, the first step is to convert that query into a vector through a process called **query embedding** as shown as Step 2 in Figure 3 in the Appendix. This involves using the same embedding model that was applied to the document chunks during preprocessing. Using a consistent model for both queries and documents ensures that their representations align in the same semantic vector space.

Once the query is embedded as shown in Step 2 of Figure 3, the system performs a similarity search against the stored chunk embeddings in the vector database as shown as Step

3 in Figure 3 in the Appendix. The goal is to find the top-K most relevant chunks—those with the highest similarity scores relative to the query vector. This retrieval step allows the system to identify content that is not just lexically similar but semantically related to the user's intent.

To measure this similarity, various metrics can be used. Cosine similarity is one of the most common, assessing the angle between vectors to determine how closely they point in the same direction. Euclidean distance considers the straight-line distance between points in the vector space, while dot product is useful for ranking results when working with normalized vectors. These mathematical approaches enable the system to efficiently surface the most relevant information from even massive datasets.

In our experiment, we set up a retrieval system that uses traditional methods (like cosine similarity and dot product similarity) to find relevant documents based on a given query. We compared the two metrics to see which worked best for retrieving relevant results. We also tested different configurations (e.g., number of retrieved documents) to find the most effective setup. This setup allows us to analyze how each component of the RAG system contributes to its overall performance and efficiency.

2.5 Question Answering

In a Retrieval-Augmented Generation (RAG) pipeline, once the most relevant chunks have been retrieved, they are combined with the user's original query to create an augmented prompt. This enriched input is then passed to a Large Language Model (LLM), which generates a response grounded in the provided context. By supplying the model with actual source material, the likelihood of hallucinated or inaccurate answers is significantly reduced, as the response is anchored in retrieved data. This approach also enables domain-specific question answering by incorporating current, proprietary, or specialized information—without the need to retrain the underlying model. As a result, RAG systems can deliver more accurate and context-aware responses tailored to specific use cases.

For generating responses, we used several Large Language Models (LLMs) like GPT-4o ("o" for "omni"), Microsoft Phi3 and DistilGPT2. These models generated answers based on the documents retrieved in the previous step.

2.6 Comparison with other architectures

2.6.1 RAG vs. Traditional Language Models: Traditional language models like GPT-3 rely solely on pre-trained data, which can become outdated and prone to hallucinations. In contrast, RAG retrieves external information in real time, grounding responses in current and relevant documents. It also adds transparency by referencing sources. However, RAG introduces extra complexity and slightly slower response times due to its retrieval step, while traditional models are faster and self-contained. (Davis, 2024)

In healthcare, RAG is highly effective because it retrieves up-to-date medical information in real time, ensuring accurate responses based on the latest research and guidelines. Traditional models like GPT-3 are limited by their static, pre-trained data, making them prone to outdated or incorrect information, especially in fast-evolving fields like medicine. RAG's real-time access to current documents provides a clear advantage in delivering reliable and timely medical insights. (Harper (2025)).

2.6.2 RAG vs. Fine-Tuning: Fine-tuning adapts a model to specific tasks by retraining on new data, which can be effective but is resource-intensive and static. RAG eliminates the need for retraining by pulling updated information from external sources at inference time. This makes it more flexible and cost-efficient for knowledge-heavy tasks, while fine-tuning remains useful for style consistency or task-specific behaviors (Devline, 2025).

In the legal field, RAG shines by offering flexible, real-time access to the latest case laws and statutes, making it highly adaptable to constantly changing legal landscapes. Unlike fine-tuning, which requires extensive retraining on static data, RAG pulls in fresh information during inference, making it more cost-effective and efficient for legal tasks that require up-to-date knowledge. (Davis, 2024).

2.6.3 RAG vs. Prompt Engineering: Prompt engineering improves output by refining how queries are phrased, but it struggles with niche or evolving topics. RAG enhances accuracy by enriching prompts with real-time retrieved data, reducing the need for memorization. While prompt engineering is lightweight and requires no infrastructure, RAG offers better scalability and factual grounding—at the cost of depending on retrieval quality.

RAG outperforms prompt engineering in customer support by delivering real-time, contextually accurate answers based on the most relevant documents. While prompt engineering helps refine input phrasing, it can't keep up with niche or rapidly changing information. RAG, however, dynamically retrieves updated content, ensuring scalable, accurate support without constant manual updates. (Jaswal, 2025).

2.6.4 RAG vs. Hybrid Approaches: Hybrid methods combine RAG with fine-tuning or prompt engineering to balance accuracy, customization, and adaptability. For example, RAG with fine-tuning boosts domain relevance, while combining RAG with prompt techniques can refine retrieval focus. These setups improve performance but add complexity and require more thoughtful system design. (Fast Data Science, 2024)

In e-commerce, RAG's ability to retrieve real-time data ensures highly relevant and personalized product recommendations, making it a great choice for enhancing customer experiences. While hybrid approaches that combine RAG with fine-tuning or prompt engineering offer customization, RAG alone is already a powerful solution that balances scalability, accuracy, and adaptability, without the complexity of additional layers.

3. EXPERIMENTAL WORK

For our experiments, we used a textbook, approximately 250 pages long, covering various topics on how neural and symbolic AI methods can be combined. Its thorough and technical content made it perfect for testing our Retrieval-Augmented Generation (RAG) model, allowing us to see how effectively the system could retrieve accurate and helpful information from complex texts.

To set up the experiment, we first divided the entire text into smaller, (e.g. easier-to-handle) chunks. We tested several chunk sizes—100, 250, and 500 tokens—to figure out which size allowed the model to best capture context and retrieve relevant information. Each chunk was then transformed into numerical embeddings using three popular methods: (1.) OpenAI's text-embedding-ada-002, (2.) Sentence transformer MiniLM, and (3.) bge-small embeddings that is a small-scale English text embedding model developed by BAAI (Beijing Academy of Artificial Intelligence) in 2024. By trying these different methods, we aimed to identify which embedding approach helped the retrieval system most effectively find useful chunks. MiniLM is a family of natural language processing (NLP) models, specifically a technique for compressing large Transformer-based language model as discussed in Wang et al. (2020).

We stored these embeddings in ChromaDB, a user-friendly vector database designed for fast and efficient retrieval. We experimented primarily with cosine similarity as our measure of how closely queries matched stored chunks but also tested dot-product similarity for comparison. To evaluate how well the retrieval worked, we used straightforward metrics like precision@k, measuring how often relevant chunks appeared among the top k results, and Mean Reciprocal Rank (MRR), which checks how quickly correct information was retrieved.

Our results show clearly that the way we set up Retrieval-Augmented Generation (RAG) has a big impact on how well language models perform, even without any fine-tuning involved. We experimented with different embedding methods, chunk sizes, and similarity metrics, and found that OpenAI's text-embedding-ada-002 combined with cosine similarity performed the best, achieving an 84% retrieval precision@5. As shown in Table 1 this was notably better than Sentence transformer embeddings (76%) and bge-small embeddings (58%). Interestingly, we also saw the smaller chunks (250 tokens each) led to more precise retrieval—about 1-2% better than larger chunks (500 tokens) in case of OpenAI and sentence transformer embeddings—likely because shorter chunks captured clearer context and reduced irrelevant details.

Table 1: Precision@5 Comparison of RAG Embedding Methods with Different Chunk Sizes

RAG Embedding Method	Chunk Size = 100	Chunk Size = 250	Chunk Size = 500
OpenAI ada v2	81%	84%	83%
Sentence Transformer MiniLM	68%	76%	74%
bge-small	44%	58%	68%

To evaluate how well the generation models worked, we used F1-score as the metric. F1-score is the harmonic mean of precision and recall, providing a balanced measure of the model performance. Precision is a measure of the number of overlapping tokens between the prediction and the ground truth, divided by the total number of tokens in the prediction. Recall, on the other hand, measures the number of overlapping tokens divided by the total number of tokens in the ground truth.

When looking at how various Large Language Models (LLM) integrated with RAG, GPT-4o consistently stood out, delivering the most accurate and relevant answers. As shown in Table 2 it achieved a high response F1-score rate of 88%, compared to 82% for Microsoft Phi-3 model and 75% for DistilGPT2. GPT-4o can process and generate text, images and audio.

Table 2: Comparison of LLM with RAG

LLM with RAG	F1-score
GPT-4o	88%
Microsoft Phi-3	82%
DistilGPT2	75%

The Mean Reciprocal Rank (MRR) is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness (Wikipedia, 2024). Metrics like Mean Reciprocal Rank (MRR) supported these findings as shown in Table 3, with GPT-4o scoring 0.86 compared to 0.79 for Microsoft Phi-3 and 0.72 for DistilGPT2. Overall, these insights underline how small tweaks in embedding selection, chunking strategy, and similarity measures can substantially enhance the practical usefulness of RAG, making AI-generated responses both more reliable and more helpful**.**

Table 3: Comparison of LLM & Mean Reciprocal Rank

LLM with MRR	MRR
GPT-4o	0.86
Microsoft Phi-3	0.79
DistilGPT2	0.72

4. CONCLUSIONS & FUTURE DIRECTIONS

We assessed several popular language models, including GPT-4o, Microsoft Phi-3, and DistilGPT2, to see which one generated the most accurate and relevant answers from the retrieved information. We did this by creating typical questions based directly on topics covered in the book. Overall, this experimental setup helped us clearly understand how factors such as chunk size, embedding choice, similarity metrics, and model type influenced the quality and usefulness of responses generated by the RAG approach. The combination of OpenAI's text embedding Ada, a 250-token chunk size, cosine similarity for retrieval, and the GPT-4o model for generation yielded the best results, highlighting that the careful selection of RAG components significantly impacts overall performance.

According to Yadav (2025), the evolution of RAG applications includes multimodal integration, highlighting the future direction of seamlessly combining text and image data in advanced AI models, as shown in Table 4.

Table 4: Evolution of RAG Applications
[Derived from Yadav (2025)]

Year	RAG Stage	Example
2020	Information Retrieval	Customer Support Systems
2022	Content Summarization	Business Reports
2024	Conversational AI	Virtual Assistants
2026	Multimodal Integration	Seamless Text and Image Data Integration

This development indicates a transition towards more context-aware and versatile AI solutions. As multimodal capabilities advance, RAG applications are expected to deliver richer and better user experiences.

5. REFERENCES

[1] Allen, C. (2024). **Mastering RAG Models: A Practical Guide to Building Retrieval-Augmented Generation Systems for Enhanced NLP Applications and Improved Text Generation of LLMs.** Independently published. ISBN-13: 979-8-876-05484-5.

[2] Amugongo, L. M.; Mascheroni, P.; Brooks, S. G.; Doering, S.; Seidel, J. (2024). **Retrieval Augmented Generation for Large Language Models in Healthcare: A Systematic Review**. Preprints 2024, 2024070876. https://doi.org/10.20944/preprints202407.0876.v1

[3] Amazon (2024). **What is RAG? - Retrieval-Augmented Generation Explained**. AWS. *Amazon Web Services, Inc.*, Retrieved from https://aws.amazon.com/what-is/retrieval-augmented-generation/

[4] Baker, P. (2025). **Generative AI for Dummies**. ISBN-13: 978-1-394-27074-3.

[5] Bhoyar, V. (2024). *RAG Models Decoded: From* **Theory to Practice in Retrieval Augmented Generation**. Independently published. ISBN-13: 979-8-321-65205-3.

[6] Bourne, K. (2025**). Unlocking Data with Generative AI and RAG: Enhance generative AI systems by integrating internal data with large language**

models using RAG. Packt Publishing. ISBN-13: 978-1-835-88790-5.

[7] Brener, J. (2025). **Mastering RAG for AI Agents: Build Smarter, Data-Driven AI Systems**. Independently published, ISBN-13: 979-8-307-11143-7.

[8] Chen, J., Lin, H., Han, X., & Sun, L. (2024). **Benchmarking Large Language Models in Retrieval-Augmented Generation**. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence* (AAAI'24/IAAI'24/EAAI'24), Vol. 38. AAAI Press, Article 1980, pp. 17754–17762. https://doi.org/10.1609/aaai.v38i16.29728

[9] Johnson, B. (2025). **AI for Growth.** Independently published, ISBN-13: 979-8-304-66438-7

[10] Devline, M. (2025). **Large Language Models Graph RAG: A hands-on guide to knowledge graph integration with LLMs.** Independently published. ISBN-13: 979-8-303-53605-0.

[11] Dickerson, J.A., Berleant, D., Cox, Z., Qi, W., and Wurtele, E. (2001). **Creating Metabolic Network Models Using Text Mining and Expert Knowledge**. *Atlantic Symposium on Computational Biology, Genome Information Systems & Technology (CBGIST 2001*), pp. 26-30.

[12] Ding J, Viswanathan K, Berleant D, Hughes L, Wurtele ES, Ashlock D, Dickerson JA, Fulmer A, Schnable P.S. (2005). **Using the Biological Taxonomy to Access Biological Literature with PathBinderH**. *Bioinformatics*; 21(10):2560-2562. https://doi: 10.1093/bioinformatics/bti381.

[13] A. D. Rosario, "**2024: Year of The RAG** - Predict - Medium," Medium, Dec. 12, 2023. https://medium.com/predict/2024-year-of-the-rag-581f7fd423f4 (accessed Jan. 25, 2025).

[14] Fast Data Science (2024). "**Large Language Models (LLM) and NLP: A new era of AI and ML has begun**," Fast Data Science, Apr. 2024. https://fastdatascience.com/generative-ai/llm-nlp/

[15] Harper, B. (2025). **The Potential of Retrieval-Augmented Generation (RAG) in healthcare.** https://www.oak-tree.tech/articles/rag-primer-healthcare.

[16] Davis, L. (2024). **Ways How Retrieval Augmented Generation Can Streamline Legal Research**, https://metapress.com/4-ways-how-retrieval-augmented-generation-can-streamline-legal-research

[17] Jaswal, A. (2025). **RAG in Customer Support: Enhancing Chatbots and Virtual Assistants**. https://www.signitysolutions.com/blog/rag-in-customer-support

[18] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T. et al. (2020). **Retrieval-augmented Generation for Knowledge-intensive NLP Tasks**. In NeurIPS 2020.

[19] Lisztin, P. (2025). **LLM Engineer's Handbook: Master the art of engineering large language models from concept to production.** Packt Publishing. ISBN-13: 978-1-836-20007-9.

[20] Lorica, B. (2023). **Best Practices in Retrieval Augmented Generation**, *Gradient Flow*. https://gradientflow.substack.com/p/best-practices-in-retrieval-augmented

[21] My Scale (2024). **3 Ways RAG Enhances Recommendation Systems for Personalization**. https://myscale.com/blog/rag-enhances-recommendation-systems-personalization/

[22] Nucci, A. (2024). RAG vs Fine Tuning LLMs: The Right Approach for Generative AI, **Aisera: Best Generative AI Platform for Enterpris*e*,** Jan. 20, 2024. https://aisera.com/blog/llm-fine-tuning-vs-rag/

[23] Rothman, D. (2025). **RAG-Driven Generative AI: Build custom retrieval augmented generation pipelines with Llama Index, Deep Lake, and Pinecone**. Packt Publishing, ISBN-13: 978-1-836-20091-8.

[24] Siriwardhana, S., Weerasekera, R., Wen, E., Kaluarachchi, T., Rana, R., & Nanayakkara, S. (2023). **Improving the Domain Adaptation of Retrieval Augmented Generation (RAG) Models for Open Domain Question Answering.** *Transactions of the Association for Computational Linguistics* 2023; 11 1–17. doi: https://doi.org/10.1162/tacl_a_00530

[25] Smith, J. (2025). **RAG Generative AI: A Practical Guide to Building Custom Retrieval-Augmented Pipelines and Enhancing AI Systems.** Independently published. ISBN-13: 979-8-346-17325-0.

[26] Sprinter, C. (2025). **Graph RAG LLM: Hands-on Guide to graph-based approach to retrieval-augmented generation (RAG) for Developers**. Independently published. ISBN-13: 979-8-344-36974-7.

[27] "**Retrieval augmented generation (RAG) explained [+ examples]** | Super Annotate,": Retrieved from //www.superannotate.com/blog/rag-explained.

[28] Taylor, R. (2025). **Graph RAG for AI: The essential blueprint for smarter retrieval, reasoning &**

knowledge graphs (The AI Architect's Guide: Graph-Based retrieval, AI agents, and advanced generative AI*)*. Independently published. ISBN-13: 979-8-309-82550-9

[29] Wikipedia (2024). **Mean reciprocal rank**. Retrieved from https://en.wikipedia.org/wiki/Mean_reciprocal_rank

[30] Yadav, B. (2025). **Future Trends in Retrieval-Augmented Generation: What to expect in 2025 and beyond**. Retrieved from https://www.chitika.com/future-trends-in-retrieval-augmented-generation-what-to-expect-in-2025-and-beyond.

[31] Wang, W., Wei, F, Dong, Li et al. (2020). MiniLM. **Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers**. https://arxiv.org/abs/2002.10957

APPENDIX

Figures 2 and 3 are shown below in horizontal format for clarity.

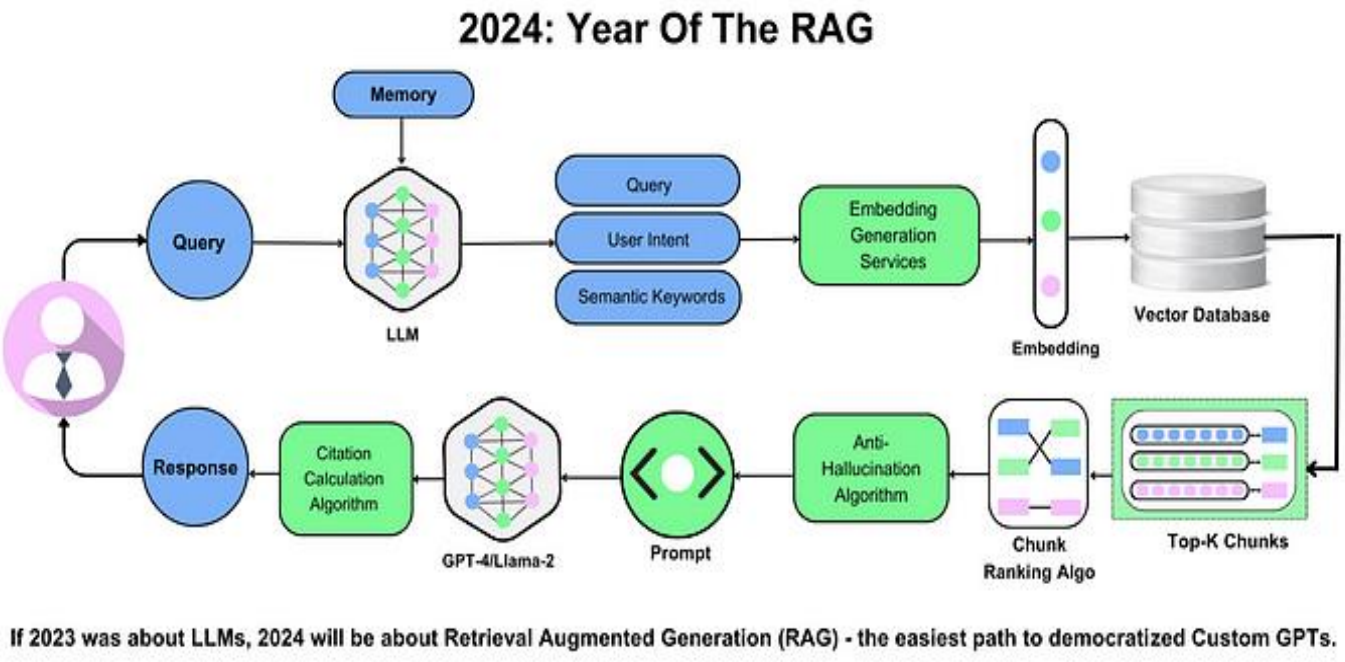

Figure 2: 2024: Year of the RAG
[Source : De Rosio (2024) https://medium.com/predict/2024-year-of-the-rag-581f7fd423f4]

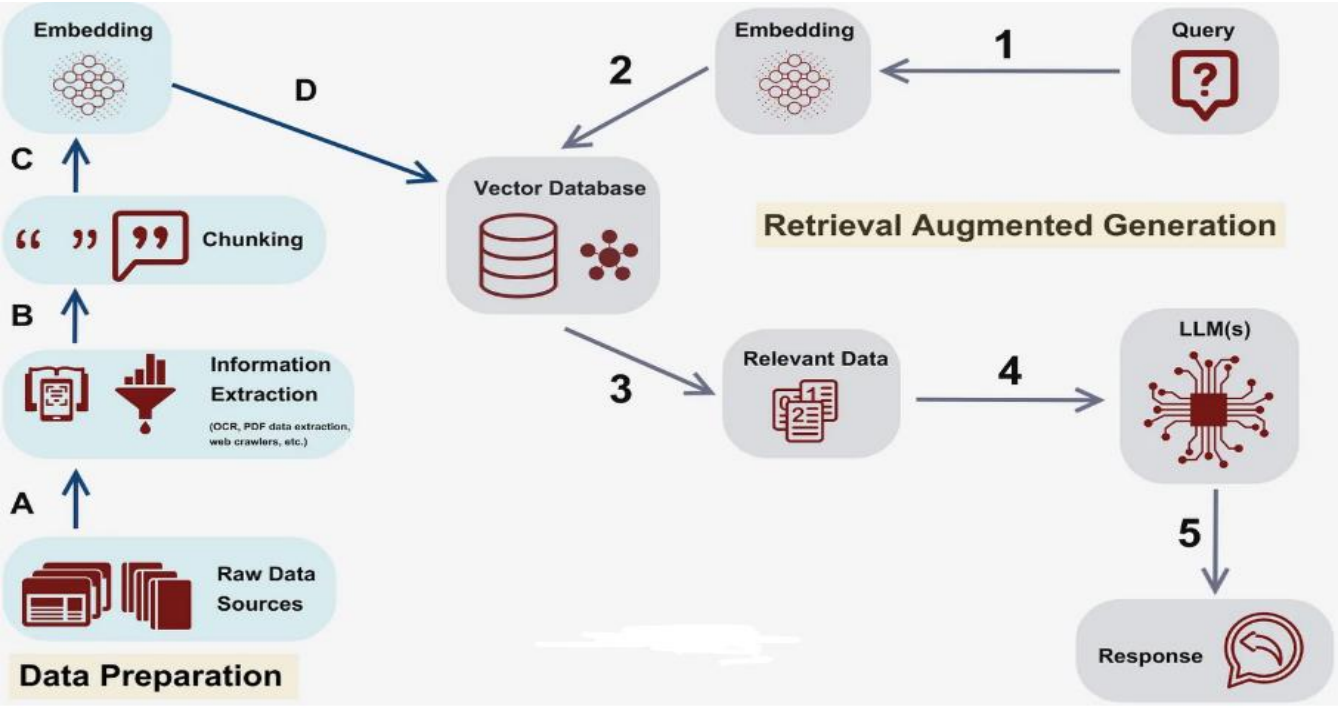

Figure 3: Data Preparation including Chunking and Steps for Responses for a RAG Application
[Source : Lorica (2023). https://gradientflow.com/wp-content/uploads/2023/10/newsletter87-RAG-simple.png]