Building Multimodal Generative AI and Agentic Applications

Shaping concept to code for the future of multimodal and advanced agentic GenAI applications

Indrajit Kar

www.bpbonline.com

First Edition 2026

ISBN: 978-93-65898-385

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My parents, my wife, my kids, and the mother

About the Author

Indrajit Kar is a distinguished AI thought leader, author of 5 AI/ML books, innovator, and author with over 22 years of experience driving transformative AI-led products and platforms across industries. Throughout his career, he has led numerous high-impact teams responsible for developing end-to-end solutions in AI, ML, GenAI, and data science - guiding projects from conceptualization and design to deployment and scaling.

In his current role as head of AI, Indrajit spearheads large-scale initiatives that deliver measurable business impact across a diverse portfolio of global clients. His work is rooted in deep technical expertise across GenAI, **large language model** (**LLM**) architectures, MLOps, natural language processing, and computer vision. He has played a key role in integrating LLMs and autonomous AI agents into real-world applications spanning sectors such as e-commerce, healthcare, life sciences, telecommunications, and manufacturing.

Indrajit is also a strategic advisor and collaborator to C-level executives, helping enterprises unlock business value through advanced AI product and platform transformations. His leadership consistently bridges the gap between cutting-edge research and enterprise-scale implementation, accelerating AI adoption across organizations.

A recognized voice in the AI community, Indrajit has authored two books, including one dedicated to GenAI and its industry applications. He has also contributed extensively to AI research, with 27+ published papers, 21 patents filed, and multiple accolades, including eight Best Paper Awards from reputed conferences and institutions. His work often explores the intersections of innovation, scalability, and responsible AI.

With a legacy of leading R&D programs and having managed AI services and productization efforts for Fortune 500 companies, Indrajit continues to shape the future of intelligent systems. His passion for innovation, combined with a vision for ethical and scalable AI, drives his mission to empower businesses and communities through transformative technology.

About the Reviewers

- **Dhanveer Singh** is a technology leader at Capital One USA with over 19 years of experience in software engineering, cloud architecture, and large-scale system modernization across financial services, insurance, and retail. He specializes in AWS, microservices, containerization, DevOps, big data, and AI/ML, delivering secure, high-performing platforms that process billions of transactions and serve millions worldwide.

  An advocate of cloud-native architectures and automation, Dhanveer has led transformative initiatives in cloud cost optimization, resilience engineering, and cybersecurity automation, driving measurable efficiency and advancing enterprise digital transformation. He has also filed multiple patents in areas of data integration, transformation, data security, and cloud automation, underscoring his focus on innovation.

  Beyond his technical leadership, he contributes as a reviewer and TPC member for international journals and conferences, serves as a judge for global IT and cybersecurity awards, and mentors through STEM and CodeDay programs.

  Dhanveer is a Fellow of IETE and IAENG, and an active IEEE and ACM member.

- **Harvendra Singh** is a distinguished technology leader specializing in cloud engineering, architecture, automation, and AI-powered solutions. He designs and implements scalable, secure systems utilizing Azure, .NET, C#, Python, GCP, Kubernetes, Databricks, and other cutting-edge technologies. With expertise in cloud-native applications, microservices, event-driven architectures, and distributed systems, Harvendra drives innovation in cloud and AI ecosystems, delivering high-impact solutions that drive business value and sustainable growth.

- **Manish Jain** is the vice president and head of AI architecture at Firstsource Solutions, where he leads enterprise-wide AI transformation for Fortune 100 organizations. With more than 20 years of technology leadership, including over a decade driving advanced AI innovation, he has earned recognition as an architect of transformative solutions that deliver quantifiable business impact. In addition to corporate responsibilities, his acts as a technical consultant for Deeplearning.ai and mentors at Analytics Vidhya. He also serves as a manuscript reviewer for prominent AI publishers such as Manning

and Packt, positioning him at the crossroads of research and practical enterprise applications. Manish is unique blend of deep technical expertise and proven executive leadership enables him to guide organizations through the strategic and operational aspects of AI transformation. His commitment to advancing the AI community is evident in his advisory and mentoring roles, as well as his involvement in peer-reviewed publishing.

These experiences make a compelling authority on the imperatives of AI transformation and the practical challenges of scaling AI across complex enterprise environments, consistently linking innovation with measurable outcomes.

Acknowledgement

I extend my deepest appreciation to my family, parents, wife, in-laws, and children, whose steadfast encouragement and belief in me have been the cornerstone of this journey. Heartfelt thanks to BPB Publications for their patience and trust, allowing the book's multi-part publication to thoroughly cover the dynamic field of AI. I am also grateful to my companies for fostering growth and providing opportunities to develop GenAI and agentic applications, which informed the insights shared here. To everyone who supported me, seen and unseen, your guidance and encouragement have profoundly shaped this journey, for which I am eternally thankful.

Preface

We are living in the age of intelligent collaboration, where AI is no longer just a tool, but a partner capable of retrieving knowledge, generating ideas, reasoning through problems, and interacting across modalities like text, images, and voice. The emergence of multimodal and agentic applications marks a turning point in how we build, deploy, and rely on AI.

This book, *Building Multimodal Generative AI and Agentic Applications*, is a practical guide for those who want to move beyond theory and actually build the future of AI systems. Across 18 chapters, you will move step-by-step from fundamentals to advanced implementations, starting with retrieval, generation, and orchestration; progressing into multimodal workflows that combine text, images, and voice; and then advancing toward real-world applications like text-to-SQL systems, OCR, fraud detection, and AI operations.

Every chapter is designed to be hands-on and approachable. You will find conceptual explanations, system design principles, code walkthroughs, and to do exercises that push you to experiment and learn by doing.

The goal of this book is not only to explain how these systems work, but also to empower you to build your own scalable, multimodal, and agentic AI applications, applications that are reliable, safe, and impactful.

Whether you are an engineer, researcher, or leader in technology, I hope that this book equips you with the knowledge, confidence, and inspiration to shape the next-generation of AI.

Chapter 1: Introducing New Age Generative AI - This chapter introduces the key building blocks of modern AI systems. It begins with an overview of generative AI and then explores retrieval systems, generation systems, and the strengths of each. It covers how **retrieval-augumented generation** (**RAG**) generation combines the two, and how orchestration helps different AI components work together. The chapter also explains tokens, vector databases, and reranking methods, along with the differences between bi-encoders and cross-encoders. Finally, it discusses essential topics like guardrails for safe AI use, the role of agents, and the importance of Model Context Protocols.

Chapter 2: Deep Dive into Multimodal Systems - This chapter focuses on vision-language models and their role in multimodal AI. It explains what vision-language models are, compares different implementation approaches, and explores how they differ from broader multimodal GenAI systems. The chapter also looks at vision-language models in more depth and introduces ways to classify multimodal systems based on their outputs.

Chapter 3: Implementing Unimodal Local GenAI System - This chapter explores the practical side of building GenAI systems. It begins with the role of GPUs in today's AI landscape and how to make use of a local GPU. The chapter then introduces Ollama, including how to generate a PDF document with it. Moving forward, it explains how RAG works, along with the key challenges involved in implementing RAG effectively.

Chapter 4: Implementing Unimodal API-based GenAI Systems - This chapter provides a hands-on introduction to working with OpenAI's APIs and models. It explains how to move from using OpenAI for basic tasks to building more advanced agentic AI solutions. You will learn how to perform multi-document queries, implement a modular retrieval-augmented generation system using OpenAI and Faiss, and explore a set of to do steps for extending these capabilities further.

Chapter 5: Implementing Agentic GenAI Systems with Human-in-the-loop - This chapter focuses on designing and advancing agentic generative AI systems. It starts with principles of architecting such systems and then walks through an end-to-end **human-in-the-loop** (**HITL**) RAG workflow. From there, it explores how HITL setups can evolve into multi-agent HITL RAG systems. The chapter concludes by clarifying the differences between agentic AI and AI agents, highlighting their distinct roles and applications..

Chapter 6: Two and Multi-stage GenAI Systems - This chapter provides a deep understanding of the concepts of interactions within dense retrieval systems and their importance in RAG. It explains the role of interaction models in two-stage RAG systems and compares different reranking strategies, including late interaction, full interaction, and multi-vector models. The chapter then introduces two-stage and multi-stage RAG architectures, discusses grading mechanisms for evaluating retrieved results, and demonstrates how to implement a multi-stage RAG workflow with routing for more accurate and efficient responses.

Chapter 7: Building a Bidirectional Multimodal Retrieval System -This chapter introduces multimodal systems and how they can be classified based on their outputs. It then explains the working of a multimodal retrieval system and provides a code implementation with step-by-step explanation. The chapter closes with a to do section, giving readers practical exercises to apply and deepen their understanding.

Chapter 8: Building a Multimodal RAG System - This chapter focuses on practical approaches to generation and evaluation using LLMs. It begins with the implementation of generation techniques, followed by an introduction to the concept of LLM-as-a-judge and its application in building recommender systems. The chapter also covers how to incorporate grading mechanisms with OpenAI to improve evaluation. It concludes with a to do section, giving readers exercises to apply these ideas in practice.

Chapter 9: Building GenAI Systems with Reranking - This chapter explores the concept of reranking and its critical role in improving retrieval and RAG systems. It explains how reranking is applied in both text-based and multimodal contexts, with a focus on using cross-encoders in multimodal RAG. The chapter also introduces the cross-encoder architecture in multimodal settings and the idea of multi-index embedding within RAG systems. Alongside these concepts, it provides a code implementation with detailed explanation and concludes with a to do section to help readers practice and solidify their understanding.

Chapter 10: Retrieval Optimization for Multimodal GenAI - This chapter examines how to make retrieval systems more efficient and effective. It begins by outlining common drawbacks of retrieval systems, then introduces various optimization techniques to address these limitations. The chapter also explores retrieval optimization in detail, showing how these methods can be applied to improve performance. It then shifts focus to multimodal RAG systems, explaining how adaptive index refresh can enhance their accuracy and responsiveness. Finally, it provides a to do section with exercises for readers to apply these ideas in practice.

Chapter 11: Building Multimodal GenAI Systems with Voice as Input - This chapter explores how RAG extends beyond just image and text. It introduces the core concepts of expanding RAG to other modalities and shows how speech interfaces can be integrated into the RAG architecture. The chapter also provides a step-by-step code implementation of a voice-enabled RAG system, demonstrating how to bring these ideas into practice.

Chapter 12: Advanced Multimodal GenAI Systems - This chapter highlights the importance of reasoning in GenAI systems. It explains the different types of reasoning used in GenAI and why they matter for building more reliable and intelligent models. The chapter also introduces key benchmarks that are used to evaluate reasoning capabilities in AI systems.

Chapter 13: Advanced Multimodal GenAI Systems Implementation - This chapter focuses on how reasoning can be enhanced in GenAI through effective prompting techniques. It then explores specialized architectures that bring reasoning into play at different stages—first during reranking, where results are refined, and then at the recommendation stage, where reasoning helps deliver more accurate and context-aware suggestions.

Chapter 14: Building Text-to-SQL Systems - This chapter delves into the complexities of text-to-SQL and why it is considered a challenging problem. It begins by explaining the basic concepts and then explores real-world applications where text-to-SQL can make a significant impact. The chapter discusses the key challenges involved, followed by practical guidance on designing an effective text-to-SQL system. It also covers entity extraction using large language models, highlighting how this integrates with text-to-SQL to improve performance. Finally,

the chapter emphasizes how such systems can enhance data accessibility and literacy, while also introducing performance metrics and best practices to ensure reliability.

Chapter 15: Agentic Text-to-SQL Systems and Architecture Decision-Making - This chapter presents the design and implementation of an agentic text-to-SQL system tailored for real-time retail intelligence. It explains the system's architecture in detail, along with code walkthroughs for better understanding. A step-by-step pipeline is provided to show how the system processes queries, leading to meaningful outputs. The chapter concludes by demonstrating the actual results generated by the text-to-SQL system and how they address the original problem statement.

Chapter 16: GenAI for Extracting Text from Images - This chapter introduces three different approaches to applying GenAI for optical character recognition. It explains how OCR works on images, as well as how it can be extended to multimodal documents that combine text, images, and other elements. The chapter concludes with a to do section, giving readers practical exercises to apply and reinforce what they have learned.

Chapter 17: Integrating Traditional AI/ML into GenAI Workflow - This chapter explores how traditional machine learning models can be integrated into GenAI workflows through a detailed case study. It presents a practical use case of hybrid ensemble learning for telecom fraud detection, showing how models like XGBoost can be wrapped and enhanced within an LLM-powered system. The chapter also provides a comparative overview of different ways ML models can be combined with GenAI to create hybrid solutions. It concludes with a to do section, offering readers hands-on activities to deepen their understanding.

Chapter 18: LLM Operations and GenAI Evaluation Techniques - This chapter highlights the importance of operations in building and running production-grade GenAI applications. It compares evaluation methods for LLMs and RAG systems, introduces the concept of RagOps, and emphasizes the need for continuous monitoring and observability platforms. The chapter also explores how graph-enhanced RAG can improve recommendation systems and provides a comparison of different Ops practices in modern software development. Finally, it offers practical guidance on setting up MLflow for managing experiments and deployments.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/78f896

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Building-Multimodal-Generative-AI-and-Agentic-Applications.
In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at: **errata@bpbonline.com**

Your support, suggestions and feedback are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our Discord space

Join our Discord workspace for latest updates, offers, tech happenings around the world, new releases, and sessions with the authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1

Introducing New Age Generative AI

Introduction

This chapter sets the stage for mastering new age **generative AI** (**GenAI**) systems by introducing essential concepts and foundational technologies. We begin by exploring the difference between retrieval systems and generation systems, followed by an in-depth look at vector databases, search algorithms, embedding techniques, indexing, and reranking, all critical for building intelligent, efficient AI solutions. Key reliability mechanisms, such as reflection and guardrails, are discussed to ensure outputs remain robust and aligned with user intent.

We then dive into advanced prompting methods like **chain of thought** (**CoT**) to guide AI models through structured reasoning processes. Moving into agentic AI, the chapter covers agents, tools, reasoning, planning, and action execution, expanding into the design of multi-agent systems capable of complex, collaborative tasks. A comparative overview of **large language models** (**LLMs**), **large vision models** (**LVMs**), and emerging **large action models** (**LAMs**) is provided, along with practical insights into local model deployment and **graphics processing unit** (**GPU**) infrastructure planning.

Further, we introduce speech technologies, including **automated speech recognition** (**ASR**) and generation, and explain the critical role of memory management in agent-based architectures. Finally, we present industry standards like **Model Context Protocol** (**MCP**) and differentiate the evolving responsibilities of a GenAI developer vs. a GenAI engineer, preparing readers for advanced system design.

Structure

This chapter covers the following topics:

- Overview of generative AI
- Retrieval system
- Generation systems
- Understanding where generation systems excel
- Retrieval-augmented generation
- Orchestration in AI systems
- Tokens in AI systems
- Vector database
- Reranking
- Bi-encoders vs. cross-encoders
- Guardrails
- Agents
- Model Context Protocols

Objectives

This chapter aims to equip readers with a comprehensive understanding of the key building blocks essential for designing and deploying modern GenAI systems. By exploring concepts such as retrieval and generation systems, vector databases, embedding techniques, advanced prompting strategies, agentic architectures, and multi-agent collaboration, readers will gain a strong foundation for building intelligent, scalable AI solutions. Additionally, the chapter introduces critical topics like local model deployment, GPU infrastructure, speech processing, memory management in agents, and industry standards like MCPs. These foundational elements are crucial for advancing toward multimodal, reliable, and production-ready AI applications.

Overview of generative AI

The evolution of generative models represents one of the most significant paradigm shifts in AI. In the pre-**generative pre-trained transformers** (**GPTs**) era, GenAI was shaped by powerful techniques such as Boltzmann machines, **variational autoencoders** (**VAEs**), **generative adversarial networks** (**GANs**), and autoencoders. These models achieved groundbreaking results by generating unstructured data like images, audio, and even text. For instance, GANs revolutionized realistic image synthesis, while VAEs enabled probabilistic generative modeling of complex data spaces, including speech and document generation.

While impressive, these earlier systems generally focused on single-domain generation with limited ability to reason, plan, or generalize across tasks. They lacked the rich contextual understanding, dynamic reasoning, and task-driven flexibility that define modern AI experiences.

The true *paradigm shift* occurred not directly with GPT models, but with the introduction of the transformer architecture itself in 2017 (in the seminal paper *Attention Is All You Need* by *Vaswani et al.*). The transformer introduced the concepts of self-attention, parallel processing, and positional encoding, enabling models to scale massively in both size and capability, far beyond the limits of traditional **recurrent neural networks** (**RNNs**), **long short-term memories** (**LSTMs**), or **convolutional neural networks** (**CNNs**) based generative models.

Building on the transformer foundation, GPTs ushered in the era of *open-ended generation* models capable of not just recreating data but performing tasks like conversation, reasoning, summarization, code generation, and multimodal synthesis. The modern GenAI systems now exhibit semantic awareness, dynamic problem-solving, and multimodal understanding across text, images, and speech.

Several key advancements define this new age, which are as follows:

- Massive pre-training on diverse, heterogeneous datasets.
- Scaling laws showing predictable improvements with more parameters, data, and compute.
- CoT prompting techniques for guided reasoning.
- Agentic AI architectures where models not only generate but also reason, plan, and act.
- Multi-agent systems collaborating toward complex goals.
- Multimodal generation across text, vision, and audio modalities.
- Private and local deployments driven by improvements in GPU infrastructure and efficient models.

Note: The scope of this book is focused exclusively on new-age GenAI systems. If you seek to explore the foundations of older generative models, including Boltzmann machines, autoencoders, VAEs, and GANs, you can refer to another book authored by me and my co-author, titled "Learn Python Generative AI: Journey from Autoencoders to Transformers to Large Language Models" (published by BPB Publications). It provides a detailed walkthrough of the classical generative modelling journey leading to today's cutting-edge systems.

In this book, we move beyond classical generation, focusing on designing, building, and deploying reasoning, planning, and action-oriented GenAI—the systems that are now transforming industries, enterprises, and everyday experiences. Understanding this transition is key: *what started as data mimicry has evolved into intelligent, multimodal agents capable of augmenting and automating human thought itself.*

While generative models have evolved to create rich, human-like outputs, not all AI solutions rely solely on generation. In fact, many of the most powerful AI systems today combine retrieval with generation to ground their outputs in real-world information, improve reliability, and reduce hallucinations.

Before exploring generation strategies, it is essential to first understand retrieval systems, the backbone of how AI finds, filters, and brings relevant knowledge into the conversation. Retrieval forms a critical pillar of modern AI infrastructure, supporting tasks ranging from search engines and recommendation systems to advanced **retrieval-augmented generation** (**RAG**) pipelines.

In the next section, we will explore what retrieval systems are, how they differ from pure generative models, and why they are indispensable for building accurate, scalable, and production-grade AI applications.

Retrieval system

GenAI systems today are celebrated for their creativity and reasoning abilities, but behind many of these intelligent behaviors lies a strong foundation built on retrieval mechanisms. Retrieval is often the hidden engine that allows AI to ground its outputs in real-world knowledge, find relevant facts, and maintain coherence across conversations or tasks. To truly appreciate how retrieval has become such a critical pillar of modern AI, it is important to first understand how it evolved, from simple keyword matching to sophisticated, learning-driven, and memory-augmented techniques.

Prior to understanding modern retrieval systems, it is helpful to trace their evolution briefly, which is discussed in the following table:

Year	Milestone	Description
1970s-2000s	**Term frequency–inverse document frequency** (**TF-IDF**), **Best Matching 25** (**BM25**).	Early keyword-based retrieval methods focused on matching exact terms.
2020	**Dense passage retrieval** (**DPR**)	Introduced dense embeddings to semantically match questions and documents.
2021	Hybrid retrieval	Combined sparse (BM25) and dense (DPR) methods to improve robustness.
2020–2022	RAG	Tight integration of retrieval with generation models to enhance grounding.
2023+	In-context learning retrieval, memory-augmented retrieval.	Dynamic, reasoning-driven retrieval embedded inside LLM workflows.

***Table 1.1**: Historic timelines of retrieval systems*

With the preceding background, given in *Table 1.1*, in mind, it becomes clear that retrieval is no longer a simple lookup process; it has evolved into a dynamic, intelligent layer that actively augments the reasoning capabilities of AI systems. In the following sections, we will explore how retrieval systems work, the key components that make them powerful, and how they integrate seamlessly with generative models to build reliable, context-aware AI applications.

The foundation of modern retrieval systems can be traced back to early innovations like DPR, introduced by *Facebook AI Research (*now *Meta AI)* around 2020. DPR was a major breakthrough compared to traditional sparse retrieval methods (such as TF-IDF and BM25) because it introduced dense vector representations for both queries and documents. This allowed semantic retrieval, finding information based on meaning rather than relying purely on keyword overlap.

Dense retrieval marked a major turning point: models could now encode the meaning of a query and a document into a shared embedding space where similarity could be computed efficiently. Instead of matching exact words, dense retrieval matched concepts and ideas. However, early dense retrievers still had limitations: they sometimes retrieved irrelevant passages due to coarse semantic matching, and scaling them to millions or billions of documents required solving difficult engineering challenges around efficiency and latency.

Sparse retrieval

Sparse retrieval methods like TF-IDF and BM25 rely on matching exact keywords and term frequency statistics. While older, they remain highly effective in cases where precision is critical and queries are closely tied to specific terminology**,** such as in legal document search, scientific literature, and enterprise document retrieval, where exact matches matter more than general semantic similarity. Sparse retrieval also scales very efficiently with traditional inverted index techniques and remains a strong baseline in many real-world search systems.

Dense retrieval

Dense retrieval methods, introduced with models like DPR and **Approximate Nearest Neighbor Negative Contrastive Learning for Dense Text Retrieval** (**ANCE**), marked a major shift from sparse term-matching techniques (e.g., BM25) toward semantic vector-based retrieval. Dense retrievers excel when dealing with open-domain search, ambiguous queries, or when synonyms and paraphrases are common, for example, in customer support bots, multilingual retrieval, or semantic **frequently asked questions** (**FAQs**) matching. Dense retrieval allows systems to understand the intent behind a question, even when the exact words differ between the query and the document. The following figure shows the basic flow of semantic retrieval using a vector database:

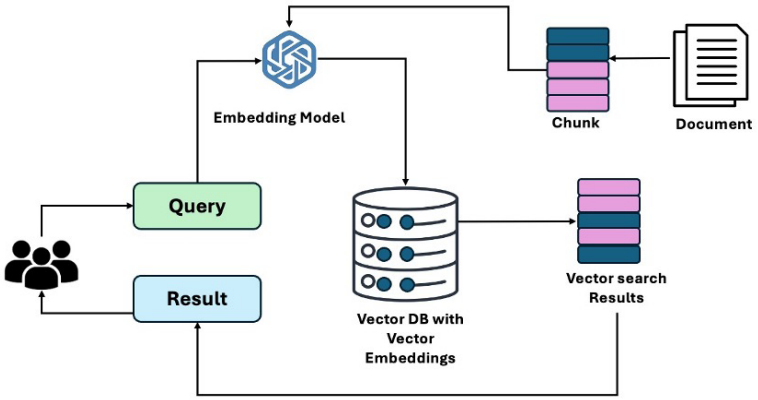

Figure 1.1: Basic flow of semantic retrieval using a vector database

Note: To maintain clarity and simplicity, this figure illustrates document chunking and embedding as part of the overall RAG process. In practice, these steps—chunking and embedding of documents- are performed offline during the indexing phase and not during real-time query execution. This simplification applies across all figures and workflows presented in the chapters of this book.

The following figure illustrates the offline phase of a RAG pipeline, where raw documents are first processed using language chunking tools (e.g., Llama-based parsers or LangChain utilities) to divide them into manageable segments. These chunks are then passed through an embedding model, such as OpenAI's embedding API, to generate dense vector representations. The resulting embeddings are stored in a vector database, forming the searchable index that powers downstream retrieval during real-time query execution. This preprocessing step is critical to enabling fast, scalable, and semantically rich document retrieval in multimodal or LLM-based applications.

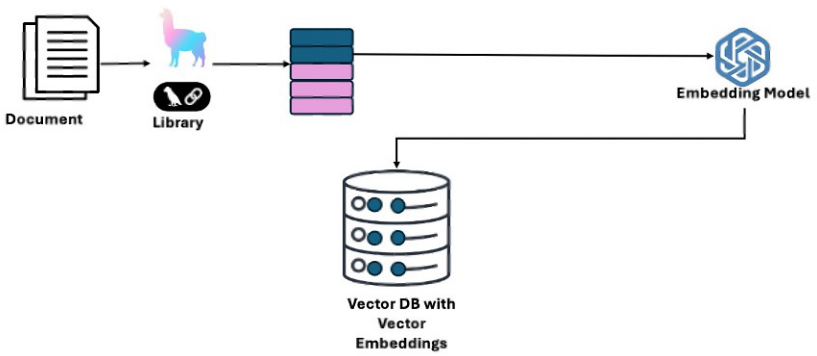

Figure 1.2: Offline document indexing and embedding workflow

Reflecting on the evolution, today's retrieval systems have dramatically advanced beyond the early DPR architecture:

- **Hybrid retrieval**: Modern systems increasingly combine sparse and dense retrieval (e.g., BM25 + dense embeddings) to balance recall and precision, especially valuable in long-tail queries or domain-specific knowledge bases.
- **Multi-vector representations**: Advanced methods like *ColBERT* (late interaction models) encode multiple vectors per document rather than a single one, improving retrieval accuracy without sacrificing too much speed.
- **Retriever-generator fusion (RAG systems)**: Retrieval is no longer a standalone step; it is now tightly integrated into the generation pipeline. Models like RAG retrieve documents dynamically during inference and condition the generated output, improving factual accuracy and reducing hallucinations.
- **Memory-augmented retrieval**: Agentic AI systems use episodic memory, blending external document retrieval with internally learned knowledge to continuously adapt and improve over time.
- **Learning-to-retrieve (LTR) and in-context retrieval**: Some newer architectures like *Retro* and *RePlug* move beyond static indexes, enabling the model itself to learn retrieval strategies during inference, deciding *what* to retrieve based on the reasoning context dynamically.

Additionally, vector database technology has matured rapidly. Tools like **Facebook AI Similarity Search** (**Faiss**), Milvus, Qdrant, Azure AI Search, and Pinecone offer scalable, high-speed vector search, supporting billions of embeddings with **approximate nearest neighbor** (**ANN**) algorithms, metadata filtering, and hybrid retrieval capabilities—all critical for powering modern enterprise-grade RAG systems.

It is crucial to recognize that retrieval today is no longer just about fetching documents. It has become an intelligent augmentation mechanism, involving filtering, reranking, reasoning, and dynamic knowledge grounding. Retrieval is evolving from a backend lookup service into a frontline reasoning component of next-generation AI.

Thus, understanding retrieval deeply, not simply as a search technique but as an intelligent augmentation strategy, is essential for building reliable, scalable, and goal-driven new-age GenAI applications.

Retrieval systems are typically evaluated based on metrics like recall@k, precision@k, and **Mean Reciprocal Rank** (**MRR**), which measure how effectively the system retrieves relevant documents among the top results. We will cover retrieval evaluation in greater detail later, but for now, it is important to remember that retrieval quality is judged by both accuracy and ranking efficiency.

Generation system

As we have seen, retrieval systems focus on finding the most relevant existing information. However, many real-world tasks demand more than just retrieval—they require creation, reasoning, and original synthesis. This is where generation systems come into play.

In this section, we will explore what generation systems are, how they operate, and the core techniques that power them. We will discuss different types of generation tasks, such as text, image, and audio creation, and understand key mechanisms like autoregressive modeling, diffusion models, and sampling strategies. Additionally, we will cover important concepts like temperature control, prompt design, and the balance between creativity and factuality.

We will also examine the typical challenges faced by generation systems, such as hallucination, coherence issues, and safety risks, and highlight where these systems truly excel, especially in tasks that demand open-ended creativity or complex problem-solving. Finally, we will briefly introduce how retrieval and generation are increasingly being combined in modern AI architectures to build more grounded and intelligent systems.

Let us begin by understanding the fundamental nature of generation systems and how they differ from purely retrieval-based approaches.

Generation systems are AI models designed to produce new content, rather than simply retrieve it. They can generate text, images, audio, code, and even multimodal outputs by learning complex patterns from training data. Unlike retrieval, which surfaces information that already exists, generation enables models to compose new sentences, invent new images, and solve new problems dynamically at inference time.

Modern generation systems are typically large-scale neural networks or LLMs trained with billions of parameters on massive datasets across multiple domains. The following figure shows the types of LLMs and generation models:

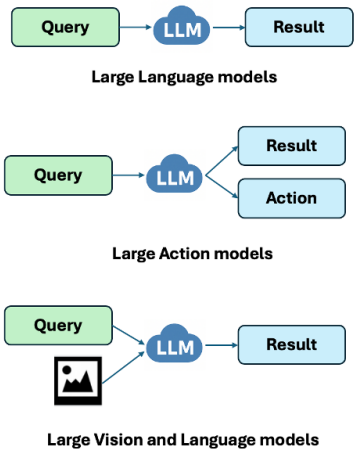

***Figure 1.3**: Types of LLMs and generation models*

Types of generation systems

GenAI systems span multiple modalities, each designed to create content such as text, images, or audio based on user input, showcasing the versatility and power of modern **machine learning** (**ML**) models. Let us look at the types of generation systems:

- **Text generation**: Models like GPT, Llama, and Claude specialize in generating coherent paragraphs, answering questions, summarizing articles, translating languages, or even writing poetry and code. They are autoregressive, meaning they predict the next token based on previous tokens—enabling them to build long, meaningful sequences word by word.
- **Image generation**: Models like DALL·E, Stable Diffusion, and Imagen generate images from text prompts (text-to-image generation). These systems rely on techniques like diffusion models or GANs to iteratively create realistic images from random noise, conditioned on user instructions.
- **Audio generation**: In audio generation, models like Whisper (for ASR) and VALL-E (for speech synthesis) produce human-like speech or even create music. These models learn representations of sound waves and either recognize speech (ASR) or generate audio based on text inputs.

Core techniques behind the generation are as follows:

- **Language models**: Language models are trained to predict the next word (token) given a previous sequence, and so they are called **autoregressive models**, as explained in *Figure 1.3*. Large models like GPT-3/4/o3, Llama, or Claude learn contextual relationships and world knowledge through self-supervised learning, enabling diverse generation tasks such as answering questions, summarizing documents, and creative writing.
- **Vision models**: Models like DALL·E and Stable Diffusion apply transformer-like architectures to image patches or latent representations, allowing text-to-image generation. They capture the structure, style, and content of visual elements in latent spaces.
- **Diffusion models**: Diffusion models start with random noise and iteratively denoise it to create a realistic sample. Popular for generating high-fidelity images (e.g., Stable Diffusion, Imagen), they have also been adapted for audio and even 3D model generation. Diffusion models are being actively adapted for language tasks, though they are still less mature and less dominant than transformer-based models (like GPT). The field of language diffusion models is rapidly evolving, and several research efforts have shown that diffusion-based generative models can be competitive with or complementary to autoregressive language models.

Autoregressive generation

In autoregressive models (like GPT), each output token is generated one at a time, conditioned on previously generated tokens. This sequential token-by-token generation allows models to produce highly coherent outputs, but can also lead to error accumulation if not managed carefully. The following figure explains how LLM generates in an autoregressive manner (one token at a time):

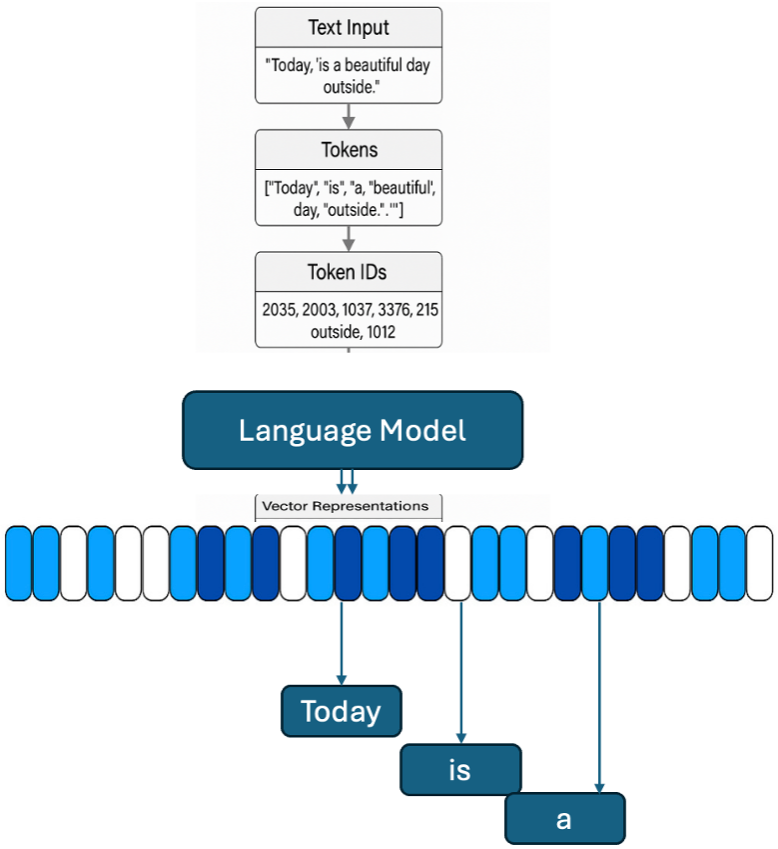

***Figure 1.4**: LLM generation in an autoregressive manner (one token at a time)*

The following are the temperature and sampling strategies:

- **Temperature**: Controls the randomness of the generation. Lower temperature | more deterministic and factual outputs. Higher temperature | more creative and diverse outputs.
- **Top-k sampling**: Limits the next token choice to the top-k most probable tokens.
- **Top-p (nucleus) sampling**: Selects from the smallest set of tokens whose cumulative probability exceeds top-p.

Tuning these parameters allows fine control over creativity vs. precision in AI generation.

Prompting strategies

Prompts are critical for steering the behavior of generation systems. Advanced prompting techniques like CoT enable multi-step reasoning by encouraging models to explain their thought process before answering. We will explain these in more detail in the next section.

Understanding where generation systems excel

Generation systems are particularly powerful in the following:

- Open-ended creativity tasks (storytelling, image creation, poetry, coding).
- Complex reasoning and problem-solving beyond retrieval capabilities.
- Personalization and dynamic response generation (chatbots, educational tutors).
- Bridging gaps where no pre-existing data exactly fits the query.

Combining retrieval and generation

While generation systems are incredibly powerful at creating new content, they sometimes struggle with factual accuracy, up-to-date knowledge, and grounding their outputs in real-world information. To overcome these challenges, modern AI architectures increasingly combine the strengths of retrieval and generation, giving rise to a powerful paradigm known as RAG.

In the next section, we will explore how RAG systems work, why they are critical for building reliable AI applications, and how they seamlessly integrate retrieval and generation into a unified, intelligent workflow.

Retrieval-augmented generation

RAG is an advanced AI architecture that combines retrieval and generation into a unified workflow. Instead of relying solely on a model's internal knowledge (which may be outdated or incomplete), a RAG system first retrieves relevant external information and then generates an answer conditioned on that retrieved content.

RAG emerged to address key challenges faced by pure generation models, which are as follows:

- **Hallucination**: It sometimes generates fabricated, plausible-sounding but incorrect outputs.
- **Stale knowledge**: Pre-trained models have a static knowledge base (cutoff dates).
- **Groundedness**: Users often demand outputs linked to verifiable, real-world information.

RAG bridges these gaps, making outputs more accurate, grounded, and up-to-date.

RAG working

A RAG system typically involves two major steps, which are as follows:

1. **Retrieval step**: Given a user query, the system first retrieves the top-k most relevant documents or chunks from an external knowledge base (e.g., a vector database).

2. **Generation step**: The retrieved documents are passed as **context** to a language model (LLM), which generates the final answer conditioned on the retrieved information.

Thus, the model is not generated from memory alone; it is reading first, then reasoning.

Architecture of a basic RAG pipeline

The following list outlines how a basic RAG pipeline looks like:

- **Query understanding**: The input query is processed, optionally rephrased or expanded, to optimize retrieval.
- **Retrieval**: A dense or hybrid retriever fetches the most relevant documents from a vector database or search engine.
- **Context preparation**: Retrieved documents are selected, truncated, chunked, and formatted to fit within the LLM's input context window.
- **Generation**: The LLM is prompted with both the original query and the retrieved documents to generate a grounded, contextually rich response.
- **Output delivery**: The model's final response is returned to the user.

Types of RAG architectures

There are many different types of RAG architectures evolving today, depending on how retrieval and generation are orchestrated. However, to keep the scope focused, the following are the two most common and practical ones:

- **Single-stage RAG**:
    - A simple pipeline: retrieve | generate.
    - Used when retrieval quality is high and latency needs to be minimal.

  The following figure shows a single-stage RAG architecture:

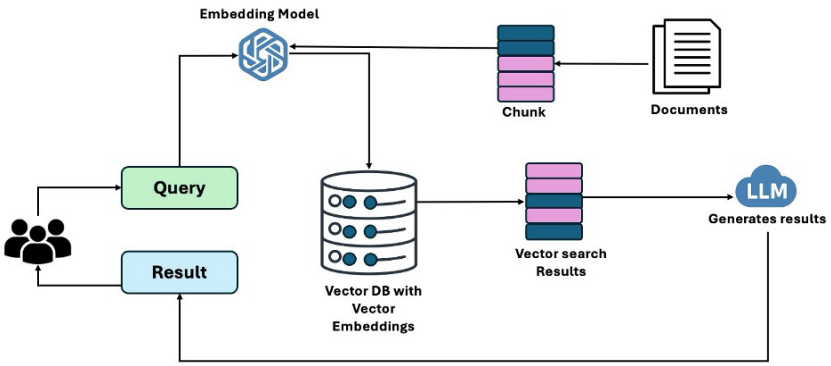

***Figure 1.5**: Single-stage RAG architecture*

- **Two-stage RAG**:
    - Retrieval | reranking | generation.
    - After initial retrieval, a second model (e.g., cross-encoder) reranks documents to improve the quality before passing them to the generator.
    - Reduces hallucination by focusing the generation only on the most relevant documents.

The following figure shows a two-stage RAG architecture:

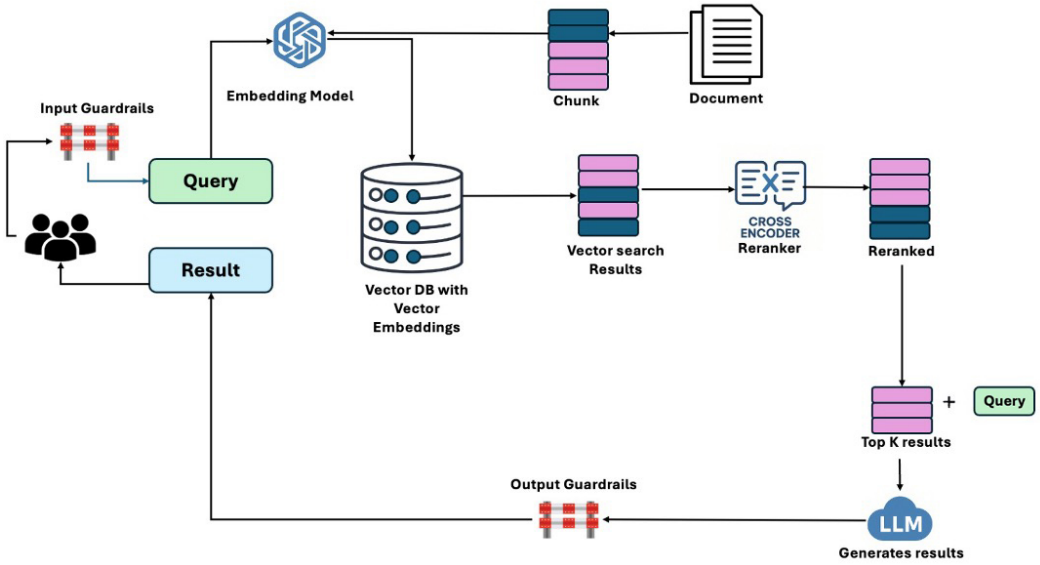

***Figure 1.6**: Two-stage RAG architecture*

Iterative RAG

The following are the two iterative RAG:

- Retrieval and generation happen across multiple turns.
- The model can retrieve additional documents dynamically if the first batch is insufficient, refining the answer step-by-step.

Vector databases and RAG

Vector databases are critical infrastructure for efficient RAG systems.

- **Purpose**: They store document embeddings and enable fast semantic search based on vector similarity.
- **Examples**: Faiss (Meta), Qdrant, Milvus, Pinecone, Weaviate.

ANN algorithms are used for scalability, finding *close enough* vectors quickly rather than exact matches, enabling real-time retrieval over millions or billions of documents.

Vector stores also allow metadata filtering (e.g., date, author) and sharding for distributed retrieval, essential for scaling enterprise RAG systems.

Prompt engineering for RAG

How the retrieved content is formatted and fed into the LLM significantly affects output quality.

Key techniques include the following:

- **Chunking**: Breaking large documents into smaller pieces to fit multiple passages into the prompt.
- **Windowing**: Sliding a fixed-size window over documents to capture local context around keywords.
- **Context management**: Selecting the most relevant chunks without exceeding the model's token limit.

Well-constructed prompts ensure the LLM focuses on the most important information during generation.

Advanced RAG techniques

As RAG systems evolve, advanced techniques are being developed to enhance retrieval quality, improve response accuracy, and enable more context-aware generation. The following are some of the advanced RAG techniques:

- **RAG with reranking**:
    - Use a reranker (like a cross-encoder) to evaluate and reorder the retrieved documents based on fine-grained relevance scoring before generation.
    - Improves precision without significantly increasing retrieval time if optimized properly.
- **Memory-augmented RAG**:
    - Retrieval is not only from static knowledge bases but also from episodic memories-storing past conversation snippets or learned experiences.
    - Enables dynamic, personalized, and context-aware responses in multi-turn dialogue systems.
- **Multimodal RAG**:
    - Extend RAG to retrieve both text and images (or videos, audio).

- **Example**: In a medical assistant role, retrieve x-rays and patient notes together, feeding both into a multimodal model like GPT-4V or Flamingo.

Applications of RAG

RAG systems have rapidly gained adoption across industries. Let us understand its applications:

- **Enterprise chatbots**: Customer service bots grounded in company knowledge bases.
- **Document QA systems**: Answering queries from large corpora like research papers, legal documents, or technical manuals.
- **Knowledge management**: Organizing and dynamically accessing enterprise knowledge in real-time.
- **Personalized AI assistants**: Tailoring responses based on user-specific documents, emails, notes, etc.

In every case, RAG ensures the AI system produces reliable, verifiable, and grounded outputs.

Orchestration in AI systems

As AI systems become increasingly complex, especially with the rise of RAG and agentic AI systems, the need for intelligent orchestration has become critical. Orchestration refers to how different components, such as retrieval engines, language models, memory modules, and external tools, are managed, sequenced, and coordinated dynamically to achieve a specific goal.

Unlike traditional single-call LLM applications, RAG systems and agentic systems involve multi-step reasoning and dynamic decision-making, requiring sophisticated orchestration frameworks.

Orchestration in RAG systems

In RAG systems, orchestration involves the following:

- **Query understanding**: Preprocessing user queries before retrieval.
- **Document retrieval**: Interfacing with vector databases (e.g., Faiss, Qdrant, Pinecone) to fetch top-k relevant documents.
- **Context preparation**: Chunking, selecting, and formatting retrieved documents to fit within the LLM's context window.
- **Prompt construction**: Dynamically inserting retrieved knowledge into well-structured prompts.
- **Response generation**: Using the LLM to generate outputs grounded in the provided documents.
- **Post-processing (optional)**: Filtering, reranking, or verifying model outputs.

Frameworks like LangChain, LlamaIndex, and Haystack specialize in orchestrating these steps automatically, making it easier to build scalable and production-ready RAG pipelines.

The following figure explains how LangChain is orchestrating the entire RAG process:

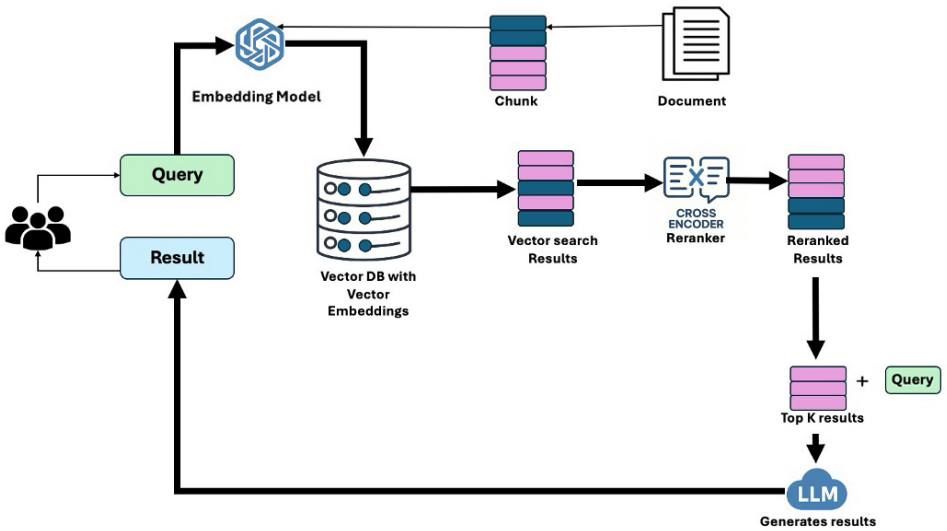

***Figure 1.7**: The fat lines are orchestrated by LangChain or similar orchestrators*

Good RAG orchestration ensures the following:

- Minimal latency
- High retrieval quality
- Tight coupling between retrieval and generation
- Robust handling of token limits and memory

Orchestration in agentic systems

In agentic systems, orchestration becomes even more dynamic.

An agent is an AI entity capable of the following:

- Reasoning about a task.
- Choosing actions (e.g., tool usage, API calls, retrievals).
- Executing actions step-by-step.
- Reflecting and adjusting its plan dynamically based on intermediate results.

Agentic orchestration involves the following:

- **Tool selection**: Deciding which external tools or functions to call based on the current goal.