

AI Agents and Applications

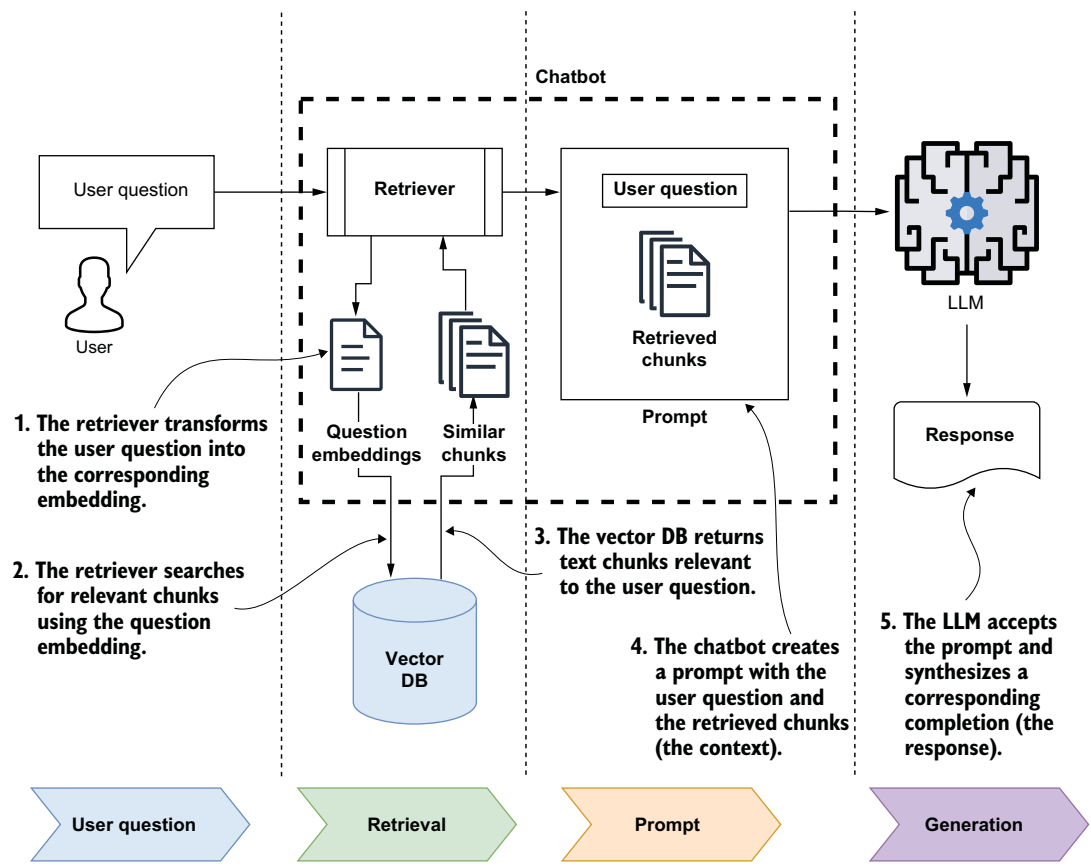
With LangChain, LangGraph and MCP

Roberto Infante



MANNING

RAG Q&A Stage



Retrieval-Augmented Generation (RAG) Q&A stage: retrieval and generation

AI Agents and Applications

AI Agents and Applications

WITH LANGCHAIN, LANGGRAPH AND MCP

ROBERTO INFANTE



MANNING
SHELTER ISLAND

For online information and ordering of this and other Manning books, please visit www.manning.com. The publisher offers discounts on this book when ordered in quantity. For more information, please contact

Special Sales Department
Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964
Email: orders@manning.com

©2026 by Manning Publications Co. All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by means electronic, mechanical, photocopying, or otherwise, without prior written permission of the publisher.

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in the book, and Manning Publications was aware of a trademark claim, the designations have been printed in initial caps or all caps.

- ⊗ Recognizing the importance of preserving what has been written, it is Manning's policy to have the books we publish printed on acid-free paper, and we exert our best efforts to that end. Recognizing also our responsibility to conserve the resources of our planet, Manning books are printed on paper that is at least 15 percent recycled and processed without the use of elemental chlorine.

The author and publisher have made every effort to ensure that the information in this book was correct at press time. The author and publisher do not assume and hereby disclaim any liability to any party for any loss, damage, or disruption caused by errors or omissions, whether such errors or omissions result from negligence, accident, or any other cause, or from any usage of the information herein.

 Manning Publications Co.
20 Baldwin Road
PO Box 761
Shelter Island, NY 11964

Development editor: Dustin Archibald
Technical editors: Keerthivasan Santhanakrishnan
and Antowan Malik Batts
Review editor: Kishor Rit
Production editor: Kathy Rossland
Copy editor: Julie McNamee
Proofreader: Melody Dolab
Technical proofreader: Andrew R. Freed
Typesetter and cover designer: Marija Tudor

ISBN 9781633436541
Printed in the United States of America

To my mother and father

brief contents

PART 1	GETTING STARTED WITH LLMs	1
1	■ Introduction to AI agents and applications	3
2	■ Executing prompts programmatically	27
PART 2	SUMMARIZATION	53
3	■ Summarizing text using LangChain	55
4	■ Building a research summarization engine	69
5	■ Agentic workflows with LangGraph	103
PART 3	Q&A CHATBOTS	119
6	■ RAG fundamentals with ChromaDB	121
7	■ Q&A chatbots with LangChain and LangSmith	143
PART 4	ADVANCED RAG	171
8	■ Advanced indexing	173
9	■ Question transformations	205
10	■ Query generation, routing, and retrieval postprocessing	228
PART 5	AI AGENTS	265
11	■ Building tool-based agents with LangGraph	267
12	■ Multi-agent systems	293

13	■	Building and consuming MCP servers	308
14	■	Productionizing AI agents: Memory, guardrails, and beyond	327
<i>appendix A</i>		<i>Trying out LangChain</i>	<i>351</i>
<i>appendix B</i>		<i>Setting up a Jupyter Notebook environment</i>	<i>357</i>
<i>appendix C</i>		<i>Choosing an LLM</i>	<i>360</i>
<i>appendix D</i>		<i>Installing SQLite on Windows</i>	<i>375</i>
<i>appendix E</i>		<i>Open source LLMs</i>	<i>377</i>

contents

<i>preface</i>	<i>xvi</i>
<i>acknowledgments</i>	<i>xviii</i>
<i>about this book</i>	<i>xx</i>
<i>about the author</i>	<i>xxv</i>
<i>about the cover illustration</i>	<i>xxvi</i>

PART 1 GETTING STARTED WITH LLMs 1

1	<i>Introduction to AI agents and applications</i>	3
1.1	Building LLM-based applications and agents	4
	<i>LLM-based applications: Summarization and Q&A engines</i>	<i>4</i> ▀ <i>LLM-based chatbots</i> 7 ▀ <i>AI agents</i> 8
1.2	Introducing LangChain	11
	<i>LangChain architecture</i>	<i>12</i> ▀ <i>LangChain's core object model</i> 15
1.3	Typical LLM use cases	19
1.4	How to adapt an LLM to your needs	20
	<i>Prompt engineering</i>	<i>20</i> ▀ <i>RAG</i> 20 ▀ <i>Fine-tuning</i> 22
1.5	Which LLMs to choose	23
1.6	What you'll learn from this book	24

2 *Executing prompts programmatically* 27

2.1 Running prompts programmatically 28

Setting up the environment for this chapter 28 ■ *Minimal prompt execution* 30

2.2 Running prompts with LangChain 31

2.3 Prompt templates 32

Implementing a prompt template with a Python function 32
Using LangChain's PromptTemplate 33

2.4 Prompt types 34

Text classification 34 ■ *Sentiment analysis* 35 ■ *Text summarization* 36 ■ *Composing text* 36 ■ *Question answering* 38 ■ *Reasoning* 39

2.5 Reasoning in detail 40

One-shot learning 40 ■ *Two-shot learning* 41 ■ *Providing steps* 41 ■ *Few-shot learning* 42 ■ *Implementing few-shot learning with LangChain* 44 ■ *Chain of Thought* 46

2.6 Prompt structure 48

PART 2 SUMMARIZATION 53

3 *Summarizing text using LangChain* 55

3.1 Summarizing a document bigger than the context window 56

Chunking the text into Document objects 57 ■ *Split* 58
Map 59 ■ *Reduce* 60 ■ *MapReduce combined chain* 60
MapReduce execution 61

3.2 Summarizing across documents 61

Creating a list of Document objects 63 ■ *Wikipedia content* 63
File-based content 64 ■ *Creating the Document list* 65
Progressively refining the final summary 65

3.3 Summarization flowchart 67

4 *Building a research summarization engine* 69

4.1 Overview of a research summarization engine 70

4.2 Setting up the project 71

4.3 Implementing the core functionality 73

Implementing web searching 73 ■ *Implementing web scraping* 74 ■ *Instantiating the LLM client* 75
JSON to Python object converter 76

- 4.4 Enhancing the architecture with query rewriting 76
- 4.5 Prompt engineering 78
 - Crafting web search prompts* 78 ▪ *Crafting summarization prompts* 81 ▪ *Research report prompt* 81
- 4.6 Initial implementation 82
 - Importing functions and prompt templates* 82 ▪ *Setting constants and input variables* 83 ▪ *Instantiating the LLM client* 83 ▪ *Generating the web searches and collecting the results* 83 ▪ *Scraping the web results* 85 ▪ *Summarizing the web results* 86 ▪ *Generating the research report* 87
- 4.7 Reimplementing the research summary engine in LCEL 89
 - Assistant Instructions chain* 91 ▪ *Web Searches chain* 93
 - Search and Summarization chain* 94 ▪ *Web Research chain* 99

5 *Agentic workflows with LangGraph* 103

- 5.1 Understanding agentic workflows and agents 104
 - Workflows* 105 ▪ *Agents* 106 ▪ *When to use agent-based architectures* 106 ▪ *Agent development frameworks* 106
- 5.2 LangGraph basics 107
- 5.3 Moving from LangChain chains to LangGraph 107
- 5.4 LangGraph core components 108
 - StateGraph structure* 109 ▪ *State management and typing* 109
 - Node functions and edge definitions* 110 ▪ *Entry points and end conditions* 110
- 5.5 Turning the web research assistant into an AI agent 110
 - Original LangChain implementation overview* 111 ▪ *Identifying components for conversion* 112 ▪ *Step-by-step transformation process* 112 ▪ *Code comparison and benefits realized* 116

PART 3 *Q&A CHATBOTS*..... 119

6 *RAG fundamentals with ChromaDB* 121

- 6.1 Semantic search 122
 - A basic Q&A chatbot over a single document* 122 ▪ *A more complex Q&A chatbot over a knowledge base* 126 ▪ *The RAG design pattern* 127
- 6.2 Vector stores 130
 - What's a vector store?* 130 ▪ *How do vector stores work?* 131
 - Vector libraries vs. vector databases* 131 ▪ *Most popular vector*

stores 132 ■ *Storing text and performing a semantic search using Chroma 133*

6.3 Implementing RAG from scratch 136

Retrieving content from the vector database 137 ■ *Invoking the LLM 137* ■ *Building the chatbot 139* ■ *Recap of RAG terminology 140*

7 Q&A chatbots with LangChain and LangSmith 143

7.1 LangChain object model for Q&A chatbots 144

Content ingestion (indexing) stage 144 ■ *Q&A (retrieval and generation) stage 146*

7.2 Vector store content ingestion 148

Splitting and storing the documents 149 ■ *Removing duplication 150* ■ *Ingesting multiple documents from a folder 151*

7.3 Q&A across stored documents 152

Querying the vector store directly 153 ■ *Asking a question through a LangChain chain 153* ■ *Completing the RAG chain setup 154* ■ *Follow-up question 156*

7.4 Chatbot memory of message history 157

Amending the prompt 158 ■ *Updating the chat message history 159* ■ *Feeding the chat history to the RAG chain 160* ■ *Putting everything together 160*

7.5 Tracing execution with LangSmith 163

PART 4 ADVANCED RAG 171

8 Advanced indexing 173

8.1 Improving RAG accuracy 174

Content ingestion stage 174 ■ *Question-answering stage 175*

8.2 Advanced document indexing 177

8.3 Splitting strategy 177

Splitting strategies 178 ■ *Factors to consider 179* ■ *Choosing the right strategy 179* ■ *Splitting by HTML header 179*

8.4 Embedding strategy 183

Embedding child chunks with ParentDocumentRetriever 184
Embedding child chunks with MultiVectorRetriever 187

Embedding document summaries 189 ▪ *Embedding hypothetical questions* 193

8.5 Granular chunk expansion 197

8.6 Semi-structured content 201

8.7 Multimodal RAG 202

9 *Question transformations* 205

9.1 Rewrite-Retrieve-Read 206

Retrieving content using the original user question 209

Setting up the query rewriter chain 210 ▪ *Retrieving content with the rewritten query* 211 ▪ *Combining everything into a single RAG chain* 212

9.2 Generating multiple queries 213

Setting up the chain for generating multiple queries 215

Setting up a custom multi-query retriever 216 ▪ *Using a standard MultiQueryRetriever instance* 218

9.3 Step-back question 218

Setting up the chain to generate a step-back question 220

Incorporating step-back question generation into the RAG chain 220

9.4 Hypothetical Document Embeddings (HyDE) 222

Generating a hypothetical document for the user question 223

Integrating the HyDE chain into the RAG chain 223

9.5 Single-step and multi-step decomposition 224

10 *Query generation, routing, and retrieval postprocessing* 228

10.1 Content database query generation 229

10.2 Self-querying (metadata query enrichment) 230

Ingestion: Metadata enrichment 232 ▪ *Q&A on a metadata-enriched collection* 234

10.3 Generating a structured SQL query 239

Installing SQLite 240 ▪ *Setting up and connecting to the database* 240 ▪ *Generating SQL queries from natural language* 242 ▪ *Executing the SQL query* 244

10.4 Generating a semantic SQL query 244

Standard SQL query 245 ▪ *Semantic SQL query* 246

Creating the embeddings 246 ▪ *Performing a semantic SQL search* 247 ▪ *Automating semantic SQL search* 247

Benefits of a semantic SQL search 247

- 10.5 Generating queries for a graph database 248
- 10.6 Chain routing 251
 - Setting up data retrievers* 252 ▀ *Setting up the query router* 252
 - Integrating the chain router into a full RAG chain* 254
- 10.7 Retrieval postprocessing 256
 - Similarity postprocessors* 257 ▀ *Keyword postprocessors* 257
 - Time weighting* 257 ▀ *RAG fusion (Reciprocal Rank Fusion)* 258

PART 5 AI AGENTS 265

11 *Building tool-based agents with LangGraph* 267

- 11.1 Starting simple: Building a single-tool travel info agent 268
 - Project setup* 268 ▀ *Loading environment variables* 269
 - Preparing the travel information vector store* 269
- 11.2 Enabling agents to call tools 271
 - From function calling to tool calling* 271 ▀ *How tool calling works with LLMs* 273
 - Registering tools with the LLM* 273
 - Agent state: Tracking the conversation* 275 ▀ *Executing tool calls* 275
 - The LLM node: Coordinating reasoning and action* 277
- 11.3 Assembling the agent graph 277
- 11.4 Understanding the agent graph structure 278
- 11.5 Running the agent chatbot: The Read-Eval-Print Loop 279
- 11.6 Executing a request 279
 - Step-by-step debugging* 280
- 11.7 Expanding your agent: Adding a weather forecast tool 283
 - Implementing a mock weather service* 283 ▀ *Creating the weather forecast tool* 284
 - Updating the agent for multi-tool support* 284
- 11.8 Executing the multi-tool agent 284
 - Running the multi-tool agent (initial behavior)* 285 ▀ *Improving LLM tool usage with system guidance* 285
- 11.9 Using prebuilt components for rapid development 288
 - Refactoring to use the LangGraph ReAct agent* 288
 - Running the prebuilt agent* 289 ▀ *Observing and debugging with LangSmith* 289
 - Enabling LangSmith tracing* 289

12 *Multi-agent systems* 293

- 12.1 Building an accommodation booking agent 294
 - Hotel booking tool* 294 ■ *B&B booking tool* 295
 - ReAct accommodation booking agent* 297
- 12.2 Building a router-based travel assistant 298
 - Designing the router agent* 298 ■ *Routing logic* 298
 - Building the multi-agent graph* 300 ■ *Trying out the router agent* 301
- 12.3 Handling multi-agent requests with a Supervisor component 302
 - The Supervisor pattern: An agent of agents* 302 ■ *From “one-way” to “return ticket” interactions* 304 ■ *Trying out the Supervisor agent* 305

13 *Building and consuming MCP servers* 308

- 13.1 Introduction to MCP servers 309
 - The problem: Context integration at scale* 309 ■ *The solution: The Model Context Protocol* 310 ■ *The MCP ecosystem* 311
- 13.2 How to build MCP servers 312
 - Essential resources for MCP server development* 312 ■ *Official language-specific MCP SDKs* 312 ■ *Consuming MCP servers in LLM applications and agents* 313
- 13.3 Building a weather MCP server 314
 - Implementing the MCP server* 314 ■ *Trying out the MCP server with MCP Inspector* 316 ■ *Consuming the MCP server from a test MCP host* 320
- 13.4 Integrating the Weather MCP tool into an agent 322
 - Preparing the travel agent for live weather data* 323
 - Integrating the AccuWeather MCP tool* 323 ■ *Updating the agent chat loop* 323 ■ *Combining local and remote tools* 324
 - Testing and verification* 324 ■ *Using the agent for complex queries* 325

14 *Productionizing AI agents: Memory, guardrails, and beyond* 327

- 14.1 Memory 328
 - Types of memory* 328 ■ *Why short-term memory is needed* 328
 - Checkpoints in LangGraph* 329 ■ *Adding short-term memory to*

	<i>our travel assistant</i>	331	▪	<i>Executing the checkpointer-enabled assistant</i>	334	▪	<i>Rewinding the state to a past checkpoint</i>	334
14.2	Guardrails	337						
	<i>Implementing guardrails to reject nontravel-related questions</i>	338						
	<i>Implementing more restrictive guardrails at the agent level</i>	343						
14.3	Beyond this chapter	346						
	<i>Long-term user and application memory</i>	346	▪	<i>Human-in-the-loop</i>	346	▪	<i>Post-model guardrails</i>	347
	<i>Evaluation of AI agents and applications</i>	347	▪	<i>Deployment on the LangGraph platform and Open Agent Platform</i>	348			
appendix A	Trying out LangChain	351						
appendix B	Setting up a Jupyter Notebook environment	357						
appendix C	Choosing an LLM	360						
appendix D	Installing SQLite on Windows	375						
appendix E	Open source LLMs	377						
	index	411						

preface

In late 2022, something changed. Large language models (LLMs) stopped feeling like experimental demonstrations and started becoming genuinely useful. A quick attempt to summarize a paragraph evolved into a chatbot capable of answering questions, and a small script turned into a service that other teams wanted to try. Before long, LLMs were no longer a curiosity—they had become an essential part of the software development toolkit.

Here's why that's so exciting: LLMs allow software to “speak human.” They can revise a contract, turn logs into meaningful answers, draft code, and plan the next step—and then invoke the right tools and data to actually accomplish the task. Combined with retrieval and tool use, an application stops feeling like a rigid machine and begins to feel more like a collaborative partner. The potential is significant, but turning that potential into production systems isn't simple. It still requires careful work to integrate data flows, design effective prompts, ground answers with retrieval, orchestrate multi-step workflows, and monitor how the system behaves once it's deployed.

My own journey into this field followed a similar path to many developers. I began by experimenting in Jupyter Notebooks, exploring APIs, and learning where the models succeeded and where they struggled. Those early explorations gradually evolved into small, agent-based side projects at work, built for a handful of early adopters seeking productivity gains. As the technology advanced—through improvements in OpenAI's APIs, the growing capabilities of LangChain, the orchestration power of LangGraph, and emerging techniques such as advanced Retrieval-Augmented Generation (RAG) and ReAct—those prototypes became more sophisticated. At the same time, I began writing this book. More than once, a chapter I had just completed felt outdated only a few weeks later. It was both exhilarating and, at times, exhausting.

That experience strongly influenced the way this book is written. Rather than chase every new parameter or short-lived “best practice,” the focus here is on the concepts, architectures, and design patterns that have proven stable and that underpin reliable LLM applications. We’ll build concrete systems—engines, chatbots, and agents—but the aim is to provide reusable foundations: how to structure retrieval, design prompts, compose chains, evaluate behavior, and orchestrate multi-step workflows with clarity and confidence.

Frameworks play a crucial role in this process. LangChain standardizes the essential components—loaders, splitters, embeddings, retrievers, vector stores, and prompts—so that you don’t need to reinvent the plumbing for every project. LangGraph extends this by structuring workflows as graphs and coordinating agent loops, while LangSmith adds visibility for debugging and evaluation. Together, they enable developers to focus on the application itself rather than the underlying infrastructure.

Why this book, and why now? The foundations are finally stable enough to teach effectively, and the need is greater than ever. Development teams want practical guidance that bridges the gap between ideas and implementation without tying them to a single vendor or a fleeting technique. If this book succeeds, you’ll come away with a clear mental model for building LLM-powered systems, a set of proven patterns you can depend on, and the confidence to keep building—even as the landscape continues to evolve beneath our feet.

acknowledgments

Writing a book like this is never a solo effort, and I feel profoundly grateful to have had so many talented and generous people alongside me on this journey. First, I would like to thank Juan-Mauro Bozzano, Romulus Corneanu, Carolyn Kao, and Ivan Lin for their thoughtful feedback at the very beginning, when the first chapters were still little more than outlines. Their comments and conversations helped shape the direction of the book in its early stages. I'm also indebted to David Bujan and Diego López-de-Ipiña from Deusto University, who encouraged me to bring greater precision and rigor to the work—especially in those formative drafts when many of the ideas were still taking shape.

A special word of thanks goes to Michael Stephens at Manning, who believed in this project from the very first conversation. At the time, LLMs were only beginning to capture the world's attention, and the road ahead was far from clear. His support, together with timely guidance and encouragement to adapt as the technology evolved, helped keep the book relevant and focused.

I'm particularly grateful to my editor, Dustin Archibald, whose steady guidance and thoughtful advice carried me through every stage of the writing process, and to Andrew Freed, whose sharp technical insights made the explanations clearer and more accurate. Many thanks also to technical editor Antowan Batts. Antowan is a financial systems analyst and applied economist with expertise in finance, data analysis, and global supply chains. He teaches economics and focuses on connecting analytical insights to real-world business challenges.

To the reviewers—Abdullah Al Imran, Abhishek Guha, Andres Mariscal, Antowan Malik Batts, Artur Guja, Aryan Jadon, Balaji Venkateswaran, Borko Djurkovic, Byron Galbraith, Carnell Greenfield, David Caswell, David Jacobs, Derek Morgan, Felipe P.