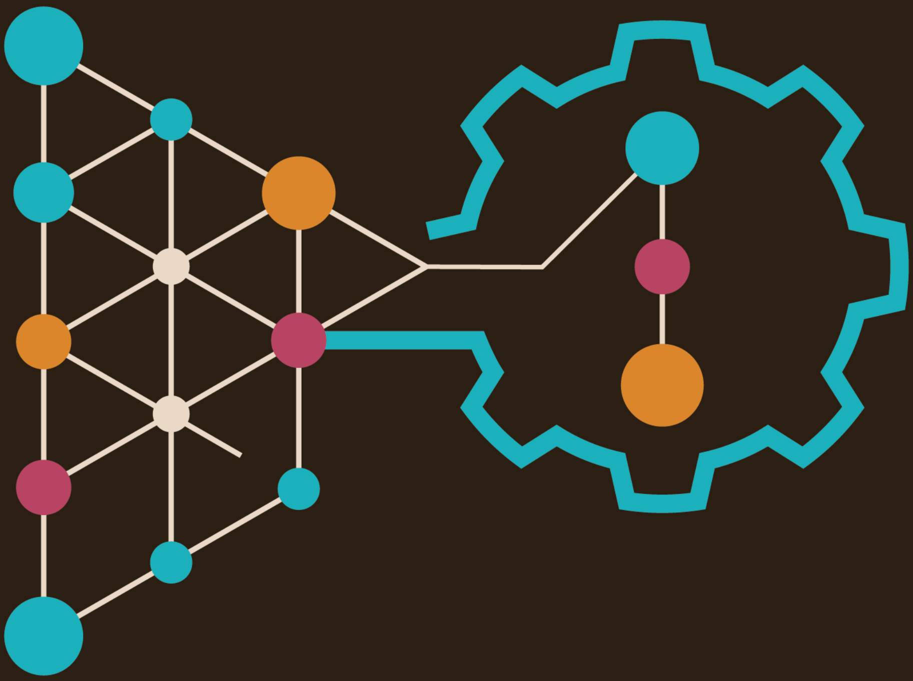
Building effective LLM-Based Applications with Semantic Kernel



Willem Meints

Building effective LLM-based applications with Semantic Kernel

Willem Meints

This book is available at
https://leanpub.com/effective-llm-applications-with-semantic-kernel

This version was published on 2025-10-15

Contents

Preface . . . . **i**
- Why I wrote this book i
- Who this book is for i
- What this book covers i
- System requirements iii
- Running the sample code in this book iv
- Feedback and issues iv

Understanding Large Language Models **1**
- What are LLMs, and why do they matter 2
- My journey with LLMs 4
- The current LLM landscape 7
- Key concepts and terminology 13
- Practical considerations for working with LLMs 18
- Real-world applications and use case 19
- Summary 21
- Further reading 21

Essential LLMOps knowledge **22**
- What is LLMOps? 22
- Testing LLM-based applications 22
- Monitoring and evaluation of your application 22
- Cost management and optimization 22
- Dealing with rate limits and capacity planning 22
- LLM performance and the user experience of your application 23
- Failover strategies 23
- Security and privacy in an LLM-based application 23
- Considerations when looking for tools 24
- Summary 24
- Further Reading 24

Getting Started with Semantic Kernel **25**
Understanding Semantic Kernel 25
Setting up your development environment 26
Setting up a project with Semantic Kernel 26
Executing your first prompt with Semantic Kernel 26
Summary 27

The art and nonsense of prompt engineering **28**
Why are prompts necessary for effective LLM-based applications? . . 28
The five basics of a good prompt 29
Writing prompt templates for reusability 30
Using the chat history to your advantage 31
Security considerations when using prompts 31
Summary 32

Testing and monitoring prompts **33**
Establishing a good test strategy for prompts 33
Using deterministic testing methods to validate prompts 33
Using model-based testing methods to validate prompts 33
The dangers of the model-based testing approach 33
Monitoring prompt interactions in production 33
Summary 35
Further reading 35

Enhancing LLMs with tools **36**
What are tools, skills, and plugins 36
When and where to use tools 36
Building tools using a kernel function 36
Sharing functions across applications with OpenAPI 37
Applying filters to functions 38
Summary 38

Retrieval augmented generation **39**
What is Retrieval Augmented Generation (RAG) 39
Building an end-to-end RAG pipeline with Semantic Kernel 39
A practical approach to validating the RAG pipeline 40
Evaluating the RAG pattern with user research 41
Variations on the RAG pattern 41
Summary 42

Working with structured output **43**
Why working with structured output is helpful 43
Applications that require structured output 43
How does structured output work under the hood 43
Getting structured output from the LLM 44
Working with a sideband channel 44
Summary 44

Prompt chaining workflows **45**
Why use a prompt chain workflow 45
Understanding and designing prompt chains 45
Building a prompt chain with Semantic Kernel 45
Testing approach for prompt chains 47
Optimizations of the prompt chain workflow 47
Summary 48

Intelligent Request Routing Workflows **49**
Why use intelligent routing in a workflow 49
Introducing to the process framework 49
Making decisions in a Semantic Kernel process 50
Building an intelligent request routing workflow 50
Considerations for using intelligent request routing 51
Summary 51

Working with agents **52**
What is and isn’t an agent 52
When to use an agent 53
Agent use cases 53
Building an agent with Semantic Kernel 53
Building multi-agent systems with Semantic Kernel 54
Testing agents 54
Security practices when working with agents 55
Summary 55

Preface

Why I wrote this book

This book came about because I needed a way to copy my knowledge to my colleagues. I almost built a custom AI agent to answer all my colleagues' questions, but it felt like giving McDonald's food to people who deserve healthy food for long-term happiness. So instead of giving quick, hallucinated together answers, I decided that my colleagues and you, as a reader, are better off with a book. I'm also gaining from this, because I structure my thoughts and get a chance to go back and improve myself.

I've been using Large Language Models (LLMs) for quite a while now and discovered how to use them to build intelligent applications by failing a lot. I can't protect anyone from making their own mistakes, but I can show what I learned and explain what worked for me.

It's challenging to write a book in this age of AI and the internet because the book is old when it hits the printing press. To combat this, I took a different approach: I'm writing and releasing it in chunks. Some chunks are lovely to read when they hit your mailbox, while others may be slightly incomplete. Please forgive my writing style and enjoy the fast-paced release cycle of the book.

Who this book is for

I wrote this book for developers who want to use an LLM in their application to solve specific challenges that can't be solved with normal program logic. All examples in the book are in C# because I think you can find plenty of Python examples online but not enough in a more enterprise-oriented language like C#.

What this book covers

- **Chapter 1, Understanding Large Language Models**, introduces large language models (LLMs) through the lens of my personal experience, explaining what they are, their capabilities, and why they're transforming how we build applications. This foundational chapter is essential for understanding the core concepts, and terminology that will help you make informed decisions about using LLMs in your projects, even if you're completely new to working with these models.

- **Chapter 2, Essential LLMOps knowledge**, explains the basic operations involved in building and hosting LLM-based applications. This chapter provides important information you'll need to know to appreciate the design patterns that we cover in later chapters.
- **Chapter 3, Getting Started with Semantic Kernel**, covers Semantic Kernel, a framework from Microsoft that simplifies building applications with LLMs by providing tools, abstractions, and patterns for integrating AI capabilities into your projects. This chapter walks you through setting up your development environment and creating your first Semantic Kernel project, giving you hands-on experience with the framework we'll build upon throughout the rest of the book.
- **Chapter 4, The art and nonsense of prompt engineering**, dives into the art of prompt engineering, teaching you how to craft effective prompts that get reliable, high-quality responses from LLMs by covering key concepts like temperature, templates, and advanced techniques. This chapter is crucial because the ability to write good prompts is the foundation of working with LLMs - without this skill, you'll struggle to get consistent results no matter what frameworks or patterns you use.
- **Chapter 5, Testing and monitoring prompts**, takes you through the steps needed to test and monitor interactions with LLMs in your application. This chapter helps you understand the importance of testing and monitoring prompts to make sure your application remains operational in the long run.
- **Chapter 6, Enhancing LLMs with tools**, explores how to enhance LLMs by giving them access to external tools, showing you how to build custom tools, integrate APIs, and manage memory and context to create more capable AI systems. LLMs become dramatically more powerful when they can interact with external tools and data - understanding these patterns will let you build AI assistants that can take actual actions and work with real-world data.

- **Chapter 7, Retrieval augmented generation**, shows you how to improve your LLM applications by grounding the responses in your own data using Retrieval Augmented Generation (RAG), going from basic vector embeddings all the way to building a working domain-specific chatbot. If you want your LLM applications to give accurate responses based on your company's documents, internal knowledge, or any specific dataset, this chapter is essential since it teaches you the complete RAG architecture from preprocessing to efficient retrieval and context integration.
- **Chapter 8, Working with structured output**, teaches you how to get structured output from LLMs, enabling you to reliably integrate AI responses into your applications. This chapter helps you bridge the gap between AI capabilities and your existing codebase. Mastering this skill will let you build reliable AI features that work seamlessly with the other components in your applications.
- **Chapter 9, Prompt chaining workflows**, delves into the essential pattern of prompt chaining, showing you how to break down complex tasks into manageable sequences of prompts that build upon each other's outputs to refine the output.
- **Chapter 10, Intelligent Request Routing Workflows**, teaches you how to build systems that intelligently route requests through a workflow by applying a reasoning prompt with logic. This chapter helps you build more flexible workflows using LLMs.
- **Chapter 11, Working with agents**, helps you understand agents and how to build them with Semantic Kernel. We'll also look at combining multiple agents to solve even more complex problems. Although agents and multi-agent systems are relatively new, this chapter will help you build a solid foundation for the future whatever it may look like with agents.

System requirements

This book contains samples in C# to demonstrate various patterns. Please make sure you have the following tools available on your system:

- .NET SDK 9.0 or Higher

- Docker Desktop or a similar containerization tool.
- Visual Studio Code or a similar code editing tool.
- Azure CLI for working with Azure OpenAI.

Running the sample code in this book

This book comes with samples on GitHub. You can find the repository at https://github.com/wmeints/effective-llm-applications/. You'll find the samples in the `samples` folder in the repository.

Each sample comes with a README.md file that explains how to run the sample and the requirements for the sample. I've made sure you can run the sample with either Azure OpenAI or the regular OpenAI service when you follow the instructions in the included README.md file.

As an added bonus, I've made sure to include samples for Java and Python too. So if you are reading this book but don't work with C#, you can still try out the samples.

Feedback and issues

If you have any feedback or issues with the book, please create an issue in the GitHub repository. I'll do my best to address the issue as soon as possible.

1: Understanding Large Language Models

I remember first realizing how useful large language models could be. Like many developers, I tried ChatGPT almost as a joke—I didn't expect it to work as well as it did. I had this piece of code that needed unit tests, and on a whim, I copied and pasted it into ChatGPT and asked it to write the tests for me. What happened next surprised me. Within seconds, I got back working code that needed only minor tweaks. The overall structure was spot-on, saving me 30 minutes of work. While 30 minutes doesn't sound like a lot of time saved, it does add up, and this is the worst-case scenario at a time when I didn't quite understand how an LLM worked.

That interaction looked like magic, but it really wasn't. It resulted from careful engineering, good quality prompt design, and understanding what LLMs can (and can't) do well. This chapter is about demystifying these powerful tools so you can put them to work in your applications.

You've probably heard the buzz around LLMs and played with ChatGPT, Claude, or other AI assistants. Perhaps you're skeptical about the hype or excited about the possibilities. Whatever your starting point, this chapter will give you the solid foundation to move beyond simple interactions and build real applications with LLMs.

We'll start with the basics – what LLMs are and why they matter – but we won't get bogged down in the theoretical. Instead, we'll focus on the practical: how these models work in the real world, what they're good at, and where they fall short. I'll share stories from my journey with LLMs, including the failures and breakthroughs that taught me the most.

By the end of this chapter, you'll understand:

- What LLMs are and why they're changing the way we solve problems
- The key concepts and terminology you need to work effectively with LLMs
- The current LLM landscape from a commercial and open-source perspective

- Practical considerations for building production applications
- Real-world use cases and applications that go beyond the obvious

Let's explore the world of large language models—not as magical black boxes but as practical tools we can understand, control, and use to solve real problems.

1.1: What are LLMs, and why do they matter

A large language model (LLM) is a neural network trained on massive amounts of data to understand and generate human-like text. It can be used for many different purposes. LLMs sound like a new thing, but they aren't.

1.1.1: A brief history

Language models aren't new at all. We've been trying to understand human language for a very long time. Early attempts involved pattern matching using regular expressions, which are well-known because you only write them once and never change them. They're hard to understand and relatively inflexible because they break at the slightest change in the input.

Later, we introduced clunky chatbots that could match patterns to understand intent using machine learning. However, these bots still had fixed responses and could not understand intent when the input was slightly off.

The real game-changers started appearing around 2017. Before that, we had specialized language models that had limited performance. Don't get me wrong, tools like SpaCy were terrific at the time for what we could do.

1.1.2: The breakthrough moment

Everything changed with the introduction of the transformer neural network architecture. The paper "Attention is all you need" demonstrated an entirely new way of processing language. I won't bore you with the technical details, but here is why it matters: Previous models would process text word-by-word and couldn't look at the context as a whole. The transformer architecture instead looks at the entire context at once, understanding the relationship between words regardless of their position.

Think about how you understand this sentence: "The developer copy-pasted the code into ChatGPT to generate unit tests for it.". You automatically know "it" refers to the code, not the tool ChatGPT. Transformers can make these connections, too, and they can do it at scale.

1.1.3: How LLMs Work

At their core, LLMs predict what comes next based on what they've seen before. When you give an LLM input text, it's not searching through a database for answers. Instead, it's using its understanding of patterns to generate responses word by word.

Here's a simplified view of what happens:

1. Your input gets broken down into tokens (pieces of words or characters)
2. The model looks at these tokens and their relationships using attention mechanisms
3. It predicts the most likely next token based on its training
4. This process repeats until it generates a complete response

The magic happens in how these models handle context. The attention mechanism helps predict the combined context of input and (the to-be-generated) output. When you feed tokens into the model, the attention mechanism state is updated with the current understanding of the context we're working on. Based on this information, the model can now more reliably predict the next likely token.

1.1.4: Why this matters for software development

You might think, "Okay, cool technology, but why should I care as a developer?" Here's why:

First, LLMs are changing how we write code. They're not just glorified autocomplete – they can understand intent. When working on a new feature, I can describe what I want in plain English, and an LLM can help me scaffold the code, suggest test cases, or point out potential issues.

So, even if you're not building AI applications, it will change how your tools work and how quickly and effectively you can write code.

Second, and this is what this book is about, they're enabling new types of applications. Think about all the tasks that were too complex or expensive to automate before because they required understanding natural language. Now, we can build applications that can:

- Generate human-like responses to customer inquiries
- Upgrade code bases from deprecated frameworks or old languages to more modern equivalents
- Translate raw input from field reports into a coherent and actionable management summary
- Help review a document that you wrote by providing valuable suggestions to improve it

And we're just getting started!

1.1.5: The current reality

Let's be clear, though – LLMs aren't magic. They have fundamental limitations. They can provide some pretty surprising responses; the response can look right but contain the wrong information, and you need to apply careful engineering to use LLMs in production systems.

That's precisely why understanding how they work is so crucial. When you know what's happening under the hood, you can:

- Design better prompts that get more reliable results
- Build safeguards against standard failure modes
- Create hybrid systems that combine LLM capabilities with traditional programming
- Make informed decisions about when (and when not) to use LLMs

In the following sections, we'll explore these practical aspects more deeply. But first, let me share some real examples from my journey with LLMs — including the mistakes I made — so you don't have to repeat them.

1.2: My journey with LLMs

My journey with LLMs started with a healthy dose of “this isn’t going to work.” Like many of you, I didn’t understand the impact of this new technology.

The unit testing experiment changed everything for me. Not because it solved all my unit-testing needs, but because it showed me that LLMs could understand context and generate meaningful, helpful output. This wasn’t just pattern matching or template filling – it was something completely different.

1.2.1: Early experiments

Once I saw what LLMs could do, I went overboard with ChatGPT. I used it for everything text-related. I even messed up two blog posts on my website with negative feedback to show for it. LLMs produce flavorless and pretty mediocre content. They’re trained to represent the average of what language has to offer. And that’s pretty average and flavorless.

I used LLMs for coding, too, as I am a developer. I wrote a complete application using only AI. And it’s used in production today. But it was hard to get there. The LLM frequently steered into the wall with weird layouts and useless unit tests. I haven’t bothered measuring how quickly I built the application. I was quicker but less satisfied with the result because writing great code is a skill I’m proud of.

After learning about open-source LLMs, I decided to try them, too. It was very slow, even on a beefy Intel Core i9 machine with a massive graphics card. I quickly learned that you need a lot of power to run an LLM on your machine and in the cloud. And with a price tag of 3500 euros for a decent machine, it’s not something you want to do for a hobby project.

There are plenty more experiences where I found the boundaries of what LLMs can do, but let me finish with one final example. I tried using an LLM to upgrade program code from a low-code solution to Typescript without human supervision. We quickly had to add human supervision because it wouldn’t help us without that.

1.2.2: Key lessons learned

You might wonder, why am I telling you my experiences? There are three key lessons that I want you to keep in mind while reading this book:

1. Be specific in what you ask from the LLM. Don't just ask for an article about LLMs; provide specific instructions.
2. Always review and understand the output of the LLM. Don't let your users use the output of the LLM unseen. The output will be wrong in all the weird ways you've never thought of.
3. Break big problems down into smaller problems. Instead of asking the LLM to perform 10 steps, ask it for just one step. It will be easier for the LLM to perform and easier for you to debug.
4. Keep track of the context and provide it in focused chunks. LLMs have limited input and output space, so they can't keep track of a complete book or even a blog post.

1.2.3: Evolution of my understanding

After the initial rollercoaster ride with LLMs, my understanding of them evolved. I stopped seeing them as a silver bullet that could solve all my language-related problems and started seeing them as a powerful pattern-matching engine capable of transforming text.

One crucial moment was realizing that LLMs excel at clearly defined tasks that involve matching a pattern in the source text and transforming it into other text. If you can find a clear pattern in the input and clearly define the target structure, an LLM is likely a good solution to your problem. The less clear the problem statement is, the more issues you'll experience.

You'll want to balance humans and machines well to build a practical LLM application. Human oversight is essential when using an LLM. Throughout the rest of the book, you will find that I'm using interaction patterns that promote human oversight because it's necessary and improves the experience significantly.

1.2.4: How I integrated LLMs into real projects

Clearly defined problems and human oversight are essential when you view an LLM-based application from a functional perspective. From a technical standpoint, you must consider applying LLMs as a software engineering problem with an AI aspect rather than a pure AI project.

Here are three reasons why you should use a software engineering approach:

- LLMs will behave better when you follow a structured approach. The more structure, the better.
- LLM behavior changes when providers push new versions of the models; automated testing is your friend.
- LLMs are slow; you must find ways to provide delayed responses to the user.
- The API endpoints of LLM providers often break, so you'll need to have solid error handling.

Throughout the rest of the book, I will share the strategies I used to maximize the use of my LLM-based applications.

1.2.5: Moving forward

These experiences shaped how I approach LLM integration today. Instead of asking, "Can an LLM do this?" I ask, "How can an LLM help with this specific aspect of the problem?" This shift in perspective has been crucial for building practical, reliable systems.

In the next section, we'll look at the current LLM landscape and how you can choose the right models for your projects. We'll build on these lessons learned to help you make informed decisions about which LLMs to use and how to use them effectively.

1.3: The current LLM landscape

After sharing my journey, let's look at the tools available to you right now. The LLM landscape is incredibly dynamic, with new models and capabilities being developed frequently. I'll focus on what's helpful in building applications.

1.3.1: Overview of major LLM providers

It's good to know that a few major LLM providers currently make the most potent models available. There are open-source options, too, but these are generally less powerful and require more engineering effort. Having said that, I recommend you try them because they offer other benefits you can't get from major LLM providers, especially the smaller models.

1.3.1.1: OpenAI

OpenAI has been at the forefront with their GPT series. What I love about OpenAI is their API's reliability. The downside? They can be expensive for large-scale applications, and their terms of service are pretty restrictive. Also, it takes a while before newer models become available through the API. If they become available, it will take a while before you can push a decent amount of input through the API because the rate limits are low.

1.3.1.2: Anthropic

Anthropic's Claude model has impressed me with its ability to handle complex instructions and longer context windows. I've found it particularly good at tasks requiring careful reasoning, like code review and technical writing. The pricing is competitive with OpenAI, and their terms of service are generally more business-friendly.

Anthropic is notorious for its rate limits. They often cap out when people in the United States are online and working, and you'll end up with random overload errors. Having a good contract with them or planning for a backup is essential.

Sadly, though, you can't use Anthropic directly with Semantic Kernel at this time, so you'll not find much about the models from Anthropic in this book. I recommend keeping an eye out for updates from Microsoft on this because I noticed that they have a few issues open on their GitHub repository that they're working on.

1.3.1.3: Meta

With LLaMA and its variants, Meta has shaken up the field by releasing powerful open-source models. While you'll need more technical expertise to use these effectively, they offer flexibility that proprietary models can't match.

LLaMA models are available through the major cloud providers, but you can also run them on your own machine if you have a good enough GPU, for example, an RTX4080 has no problem running these models. If you want to try this, I recommend looking at Ollama.

1.3.1.4: Google

Google's PaLM API and Gemini models are interesting contenders that I personally haven't had much experience with. However, if you're in the Google space, they are a great option and relatively easy to configure through their portal. I've found their documentation particularly developer-friendly.

1.3.2: Model comparison

With the major providers covered, let's break down the essential model types that I've worked with to give you an understanding of what to expect from each type.

1.3.2.1: GPT series by OpenAI

- **GPT-4o:** The king of the hill when it comes to the GPT models. This model is powerful enough for most complex tasks. It's a general-purpose model useful for code and more generic text-based tasks. This model supports interpreting images, too.
- **GPT-4o mini:** The smaller brother of the GPT-4o model is much faster while still providing plenty of capabilities. I try this model first and switch to the bigger and slower GPT-4o model when tests fail too frequently.

Recently, OpenAI started working on a new series of Orion models. These models focus on reasoning capabilities and generally lack the GPT series' general-purpose features. Currently, there are two Orion-type models:

- **o1:** This is the biggest model from OpenAI. It's at the top of biology, physics, and chemistry benchmarks. However, you're paying a premium for something that you can solve using the patterns in this book in combination with a less expensive model. Proceed with caution.
- **o1-mini:** Is the smaller version with capabilities somewhere between GPT-4o and o1. This model lacks a lot of the general knowledge included in GTP-4o and o1, so it will be useful for specific reasoning tasks but not much else.

The developments are moving fast. OpenAI already introduced a follow-up for the o1 series, the o3 series. This new series of models improves upon the reasoning capabilities of the o1 series.

You can find a complete listing of the OpenAI models on their website.

1.3.2.2: Claude models by Anthropic

Claude models come in three varieties. Just as with the GPT series, you can choose a smaller, less capable, but faster and cheaper model depending on your use case.

- **Haiku:** This is the fastest model from Anthropic. It lacks the capability to process images, but otherwise, this model provides a very fast response with good-quality reasoning capabilities.
- **Sonnet:** This is the GPT-4o equivalent from Anthropic. It's most useful for writing code and long-form content. In my personal experience, it is quite good at following more complex instructions and will stick to your style instructions quite well.
- **Opus:** This is the bigger brother of the Sonnet model. At the time of writing, it hasn't been updated since 2023, and I expect it will be a while before they update it. It is stronger than Sonnet at complex reasoning tasks.

As with the OpenAI models, I recommend testing your prompts and using the cheaper one before attempting the same task with a bigger one. Overall, the Sonnet model is your best option right now.

1.3.2.3: Gemini models by Google

Google was a little late to the game and is often considered the underdog in the LLM landscape. However, it offers various models that work well for different types of tasks. Here are the two most recent models offered by Google.

- **Gemini 1.5 Pro:** This model is the most powerful model from Google. It's a general-purpose model focusing on reasoning tasks. It works pretty well if you have a task requiring the model to generate output that meets specific constraints. This model works less for generating content like blog posts or marketing materials.

- **Gemini 2.0 Flash:** Is the fastest model offered by Google. It's much faster than the pro model but offers fewer reasoning capabilities. It's a great model for chat applications that need to be fast and only need to answer questions.

Looking at various tests and benchmarks, I've found that the Gemini models are trained mostly on instruction and constraint-solving tasks. They score lower on creative writing and coding tasks.

1.3.2.4: LLaMA and open-source alternatives

While the commercial models are powerful, the open-source models are catching up fast. Many open-source options are available at the moment, and it is hard to keep up with all the development progress. Overall, the commercial offerings provide a great overall experience. I generally start with the commercial models when exploring a use case. When I understand a use case well enough, I will try an open-source model to see whether we can lower the price point of our solution.

Having said that, I've had great experiences with these models:

- **LLaMA 3.3:** A general purpose model offered by Meta through Hugging Face. This model comes in two types: a pre-trained variant that you can fine-tune yourself to perform specific tasks and an instruction-tuned model that Meta recommends using for chat applications. But don't let yourself be limited by what Meta says because the instruction-tuned version is also useful for non-chat purposes, as long as you have an instruction-based use case.
- **Mistral:** The Mistral model by the identically named company is a fast open-source LLM mostly used for chat purposes. This model is generally less capable than the LLaMA model variants, but its speed makes up for that. This model is also hosted on HuggingFace and has many fine-tuned variants.
- **Gemma2:** Google published this model in February 2024 and trained it using a teacher/student technique. The training technique looks very interesting, but Gemma2 isn't quite as good for many of the tasks I worked on as the other models in the open-source space.

- **Phi 4:** Is a new model that was introduced by Microsoft in December 2024. It is similar to the Mistral and Gemma2 models, 14 billion parameters, but shows higher performance in the benchmarks. While this doesn't tell the whole story, it's worth trying this model for a smaller open-source model.
- **DeepSeek** is another new model that's showing great promise. The R1 variant of DeepSeek is great at reasoning and shows similar performance to the OpenAI o1 models. The V3 variant of the model is great too, for more general-purpose tasks. You should keep in mind that the DeepSeek models only support English and Chinese though, so if you need a response in other languages, your milage will vary.

1.3.3: Making practical choices

It's important to remember that the most potent model isn't always the best option. I follow this general workflow when developing an LLM-based application that can be pretty helpful if you're just starting:

1. First, I choose a general-purpose model based on the cloud provider I'm working with. Most of the time, my clients already have a contract with either Microsoft Azure or AWS to host their solutions. I use the existing environment to prototype the solution.
2. After the initial prototype, I'll examine any data privacy requirements the solution may have. Depending on these requirements, I will determine the engineering and contractual effort needed to make the solution production viable. Usually, I'm the person who talks about the technical requirements while one of our legal people looks into contracts.
3. After the initial prototype and requirements gathering, I'll deploy the solution to production for a smaller group of people to gather initial user feedback and monitor performance and costs. Based on this information, I decide whether we should optimize the solution or replace the model with something else.
4. Once the solution is optimized and running in production for an organization's general population, I'll monitor it for sudden changes in the quality of the responses or performance. We regularly test and update the models to improve the solution's overall performance.

This general workflow has helped me quite well over the past few years to deploy solutions in production. In the next section, we'll dive into key concepts and terminology for using LLMs effectively. Understanding these fundamentals will help you make better decisions during the development process of your LLM-based applications.

1.4: Key concepts and terminology

Before we dive into building applications with LLMs, let's cover some essential concepts you'll need to understand. Don't worry if some of these seem abstract at first – we'll put them into practice throughout the rest of the book.

1.4.1: Essential terminology

1.4.1.1: Tokens and tokenization

Before an LLM can process your text, it needs to break it down into tokens. Think of tokens as the building blocks the model uses to understand text. A token can be:

- A complete word
- Part of a word
- A number
- A special character
- A delimiter

The process works like this: your text gets split into tokens, then converted into numbers using the model's vocabulary. This conversion is necessary because the transformer-based neural network that powers the LLM can only process numbers, not raw text.

For example, the word "tokenization" might be split like this:

```
"tokenization" -> ["token", "ization"]
```

When the model generates a response, the process happens in reverse – tokens are converted back into text. This is why sometimes you might see slightly odd word splits in responses, especially with technical terms or rare words.

1.4.1.2: Embeddings

At the input side of almost all LLMs is something called an embedding layer. This component turns tokens into dense vectors that capture the semantic meaning of the text. These are called embeddings or embedding vectors. Embedding vectors are interesting because they can represent the relationships between words in a mathematical space.

The embedding layer isn't just a random part of the model – it's trained on vast amounts of text to understand how words relate to each other based on how they are used. Think of it as a map where similar words or concepts are located close to each other.

The embedding concept can be challenging without seeing it in motion. This website does a great job visualizing the concept.

You'll work directly with embeddings later when we implement the Retrieval Augmented Generation (RAG) pattern in Chapter 7. For now, just know they're essential for LLMs' understanding text.

1.4.1.3: Context window

Every LLM has a limit to how much text it can consider at once – this is called the context window. It's essentially the model's short-term memory, including your input and output.

Context windows have grown significantly: Most modern commercial models handle 100K-250K input tokens and around 4K output tokens. This translates to roughly 100K words on average, which is the size of a full book.

Open-source models typically have smaller windows due to the more limited number of parameters in the neural network used in these models.

The exact size is found in the model's documentation (often called a model card). As we'll see in later chapters, effectively managing this context window becomes crucial when building applications.

1.4.1.4: Output sampling and temperature

LLMs aim to produce human-like text; one way they do this is through output sampling. When generating each token, the model doesn't just pick the most likely option; it samples from a distribution of possibilities.

Temperature is your main control over this sampling process:

- Low temperature (0.1-0.3): More focused, deterministic responses
- High temperature (0.7-0.9): More creative, varied output

Here's how I typically set the temperature:

- Code generation: 0.2 (we want precision)
- Content creation: 0.7 (we want creativity)
- Factual responses: 0.1 (we want consistency)

While temperature is the most common setting you'll adjust, other sampling parameters are available. For a deeper dive into all the options, I recommend checking out this article.

1.4.1.5: Few-shot learning

Sometimes the best way to get what you want from an LLM is to show it examples. We call the use of examples in a prompt few-shot learning, and it comes in two flavors:

One-shot Learning You provide a single example:

```
Input: "The pizza was cold"
Output: Negative sentiment

Now classify: "The service was excellent."
```

Few-shot Learning You provide multiple examples:

```
Input: "The pizza was cold"
Output: Negative sentiment

Input: "The atmosphere was lovely"
Output: Positive sentiment

Now classify: "The service was excellent."
```

One good example is enough, but complex tasks might need more.

1.4.1.6: Zero-shot capabilities

Modern LLMs are so well-trained that they often perform tasks without examples. This is called the zero-shot capability of a model. You describe the structure of what you want:

```
Classify the sentiment of this review: "The service was excellent."
Give me just the score.
```

While I often start with zero-shot for simplicity, I'm not afraid to add examples if the results aren't quite what I need. The key is being flexible and pragmatic about which approach you use.

In the following section we'll examine core concepts for working with these models.

1.4.2: Core concepts

1.4.2.1: Prompt engineering

The input you give to an LLM is called a prompt. Think of it as instructions that tell the model what you want it to do. A prompt contains the task description and any context the model needs to generate the correct output.

Here's a simple example:

Bad prompt:

```
"Write a function that validates email addresses."
```

Good prompt:

```
"Write a C# function that validates email addresses. The function should:

- Use regular expressions for validation
- Return a boolean indicating if the email is valid
- Handle common edge cases like missing @ symbols"
```

You'll find many websites promoting "proven" prompt patterns, often with claims like:

- "Always start with 'You are an expert…' "
- "Use this exact format for best results…"
- "Include these specific phrases…"

I've learned the hard way that while these patterns might work initially, they often break in unexpected ways. What works better is:

- Testing your prompts thoroughly
- Adjusting temperature and other parameters
- Building robust error handling around the LLM
- Iterating based on actual usage

We'll dive deep into practical prompt engineering in Chapter 3.

1.4.2.2: Fine-tuning vs. prompt engineering

You might hear people talk about fine-tuning models for specific domains. Let me share my perspective on this.

Fine-tuning means taking a base model (like GPT-4) and training it on your specific data to improve it at particular tasks. While this sounds appealing, there are two reasons I rarely recommend it.

1. **Costs:** Fine-tuning an LLM requires significant computing resources. You also increase maintenance costs because you need to re-run the fine-tuning and validation steps each time the base model is updated.
2. **Complexity:** Fine-tuning requires in-depth knowledge of LLMs that goes beyond what we discuss in this book. You also need to collect a lot of data to fine-tune. You can use synthetic data, but it's still a lot of work.

Instead of fine-tuning, I recommend looking at alternatives like Retrieval Augmented Generation (RAG). Here's a practical example:

You're building a chatbot to answer questions about your company's products. Instead of fine-tuning a model on your product documentation, you could:

1. Store your documentation in a vector database
2. Search for relevant information when a question comes in
3. Include that information in your prompt as context
4. Let the LLM generate an answer based on the provided context

This approach is:

- More flexible (easy to update documentation)
- Cost-effective (no training required)
- Faster to implement
- More maintainable

Throughout this book, we'll explore patterns like RAG that give you the control you need without the complexity of fine-tuning. In Chapter 7, we'll implement a complete RAG system so you can see these benefits firsthand.

In the next section, we'll look at practical considerations for working with these models, building on these fundamental concepts to create reliable applications.

1.5: Practical considerations for working with LLMs

There are quite a few things that you need to account for when building LLM-based applications. This book is full of those things, but there are three cross-cutting concepts that I always keep in mind when building LLM-based applications. These are the lessons I've learned while building applications in production, and they'll help you avoid some of the nastier problems I ran into.

I think about these things as the downsides to building an LLM-based application. It's inspiring to work with language models, but they come with a cost.

1.5.1: Cost management

LLMs are resource-intensive, both in terms of computing power and API costs. If you don't need an LLM or can achieve the same result with a smaller model, I recommend doing so. If you still decide to build an LLM-based application, I highly recommend monitoring API usage and costs closely.

1.5.2: Security

Using an LLM opens up new security risks. For example, people will try to steal your application's internal instructions. They will also try to abuse your application through prompt injection. I've seen this happen in the wild, and it's not pretty. Make sure you're aware of these risks and have safeguards in place.

1.5.3: Performance

LLM-based applications can be slow, especially if you build more complicated LLM interactions. Throughout the rest of the book, you'll find strategies and patterns to help work around your performance issues. Consider this your first warning that you'll need to think about how to compensate for slow responses.

We'll explore LLMOps essentials in Chapter 2, but for now, let's look at some real-world applications and use cases to inspire you with what's possible.

1.6: Real-world applications and use case

While social media is full of posts about using LLMs for marketing content or personal task automation, the real potential goes far beyond these typical examples. Let me share some real-world cases I've worked on that showcase what's possible when you think bigger.

1.6.1: Generating targeted reports from technical information

One organization I worked with was running charity projects with donor support. They had a challenge: their field reports were too technical and detailed for donors to digest easily. Here's how we solved it:

The situation: Field reports contained crucial project information but were written in technical language that donors struggled to understand. The organization needed donor-friendly summaries but couldn't afford the time to rewrite each report manually.

We built a solution combining two key patterns: Retrieval Augmented Generation (RAG) and prompt chaining. The RAG pattern helped us find relevant information matching donor questions. In contrast, the prompt chain helped us rewrite the technical content in a donor-friendly style that matched the organization's voice.

A key learning here was the importance of human review. We had to build features showing where the information came from because reviewers wouldn't trust the system without this transparency. It's a great example of how LLMs work best when they augment human capabilities rather than trying to replace them entirely.

1.6.2: AI-Powered knowledge sharing through interviews

Another interesting case involved Info Support itself, the organization I work for. We were struggling with knowledge sharing. We had a common problem: experts too busy to write articles about their innovations, leading to repeated problem-solving across projects.

Instead of asking people to write articles, we flipped the script. We built a chat-based system where the LLM interviewed people about their work, asking progressively deeper questions to fully understand the topic. The system then transformed these conversations into structured articles.

One interesting discovery was how well users responded to AI interviews. The interaction felt natural without falling into the uncanny valley. However, we learned that letting the LLM fully control the interview flow was tricky. We eventually replaced our complex prompt-based decision-making with a more straightforward function that tracked question count to manage interview length.

1.6.3: Modernizing legacy code bases

One of my favorite projects involved upgrading legacy web forms from XML to TypeScript. The organization had so many forms that manual conversion would have taken years.

We approached this by building a batch pipeline that combined traditional XML parsing with LLM-powered code generation. Rather than asking the LLM to handle everything, we broke the problem into manageable chunks, using multiple refinement prompts to improve the generated code quality.

This project taught us a crucial lesson: LLMs work best when combined with traditional programming logic. While they're great at understanding and translating code patterns, they struggle with handling large amounts of complex code simultaneously. Breaking the problem into smaller pieces and using traditional parsing where appropriate gave us much better results.

1.6.4: We're only just starting out

These cases represent just the beginning of what's possible with LLMs. While many people start their LLM journey with personal automation through tools like ChatGPT, these examples show how to scale up to automate more complex team-based workflows.

It's important to remember that we're in the early days of this technology. The patterns I share in this book work well today, but I expect them to evolve significantly over the next few years. What excites me most is that we're just starting to understand what's possible.

As we progress in this book, we'll explore various design patterns in detail, showing you how to implement them in your projects. But first, let's make sure you have a solid foundation by discussing operating LLMs in production in the next chapter.

1.7: Summary

In this chapter, we've learned a brief history of Large Language Models and how they work from a conceptual level. We've explored the key concepts and terminology to help you understand and work with LLMs. Finally, I've shared my journey to help you understand the practical considerations and real-world applications of LLMs.

In the next chapter, we'll talk about more practical considerations you need to consider when building LLM-based applications when we discuss the essentials of LLMOps.

1.8: Further reading

Here are some resources if you want to learn more about the inner workings of LLMs:

- How LLMs think - An interesting article exploring some of the math behind LLMs in an attempt to understand how and why they work.
- Attention is all you need - The original paper that sparked the transformer revolution.

2: Essential LLMOps knowledge

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.1: What is LLMOps?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.2: Testing LLM-based applications

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.3: Monitoring and evaluation of your application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.4: Cost management and optimization

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.5: Dealing with rate limits and capacity planning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.6: LLM performance and the user experience of your application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.7: Failover strategies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.8: Security and privacy in an LLM-based application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.8.1: Data privacy

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.8.2: Application Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

2.8.3: User safety

2.9: Considerations when looking for tools

2.10: Summary

2.11: Further Reading

3: Getting Started with Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.1: Understanding Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.1.1: What is Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.1.2: Core concepts and architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.1.3: Calling Functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.1.4: Language support

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.2: Setting up your development environment

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.2.1: Other supported LLM providers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.3: Setting up a project with Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.3.1: Using Semantic Kernel in a standalone console application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.3.2: Using Semantic Kernel in an ASP.NET Core application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.4: Executing your first prompt with Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

3.5: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4: The art and nonsense of prompt engineering

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.1: Why are prompts necessary for effective LLM-based applications?

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.1.1: Why prompt engineering matters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.1.2: Common misconceptions about prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.1.2.1: Prompts are static

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.1.2.2: Complicated prompts yield better results

4.1.2.3: You can rely on the general knowledge captured in the LLM

4.2: The five basics of a good prompt

4.2.1: Provide clear direction

4.2.2: Specify the output format for the prompt

4.2.3: Add samples to the prompt

4.2.4: Keep the prompt focused on one task

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5: Tune your prompt with hyperparameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5.1: Top-P

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5.2: Temperature

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5.3: Presence Penalty

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5.4: Frequency Penalty

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.2.5.5: What to choose for each of the hyperparameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.3: Writing prompt templates for reusability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.3.1: Creating a prompt template in Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.3.2: Using Handlebars as an alternative templating language

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.3.3: Maximizing the reuse of prompts in your application

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.3.4: Using YAML-based prompt configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.4: Using the chat history to your advantage

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.5: Security considerations when using prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.5.1: Filtering executable code from prompts and the model output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.5.2: Filtering PII from the prompt and the model output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

4.6: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5: Testing and monitoring prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.1: Establishing a good test strategy for prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.2: Using deterministic testing methods to validate prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.3: Using model-based testing methods to validate prompts

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.4: The dangers of the model-based testing approach

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.5: Monitoring prompt interactions in production

5.5.1: Safety precautions when collecting telemetry

5.5.2: Enabling tracing in your LLM-based application

5.5.3: Enabling metrics in your LLM-based application

5.5.4: Logging in LLM-based applications

5.5.5: Writing monitoring data to application insights

5.5.6: Building an LLMOps dashboard with Application Insights

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.5.7: Collecting data to improve tests

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.6: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

5.7: Further reading

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6: Enhancing LLMs with tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.1: What are tools, skills, and plugins

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.2: When and where to use tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.3: Building tools using a kernel function

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.3.1: Using prompt-based functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.3.2: Creating a kernel function in C#

6.3.3: Providing functions to the kernel

6.3.4: Dependency injection in kernel functions

6.3.5: Architecting with plugins

6.4: Sharing functions across applications with OpenAPI

6.4.1: Why use external API projects as plugins

6.4.2: Setting up an API as a plugin

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.4.3: Integrating the API into your main project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.5: Applying filters to functions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

6.6: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7: Retrieval augmented generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.1: What is Retrieval Augmented Generation (RAG)

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.1.1: Retrieval component architecture

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.1.2: Storing information for retrieval

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.1.3: Retrieving relevant information

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.1.4: Generating responses

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.2: Building an end-to-end RAG pipeline with Semantic Kernel

7.2.1: Setting up the project structure

7.2.2: Building a data model for retrieval

7.2.3: Preprocessing Content into vector data

7.2.4: Using the vector store with a prompt

7.2.5: Using the vector store as a tool

7.3: A practical approach to validating the RAG pipeline

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.3.1: Overview of the validation process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.3.2: Generating validation data

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.3.3: Generating test samples

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.3.4: Measuring faithfulness

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.4: Evaluating the RAG pattern with user research

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

7.5: Variations on the RAG pattern

7.5.1: Using a Graph RAG

7.5.2: Optimizing retrieval using reranking

7.5.3: Differentiating matching and retrieval

7.6: Summary

8: Working with structured output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

8.1: Why working with structured output is helpful

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

8.2: Applications that require structured output

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

8.3: How does structured output work under the hood

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

8.3.1: Instruction-based JSON generation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

8.3.2: Constrained decoding based JSON generation

8.4: Getting structured output from the LLM

8.5: Working with a sideband channel

8.5.1: Using tool calling to get structured output

8.5.2: Integrating sideband communication in chat scenarios

8.6: Summary

9: Prompt chaining workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.1: Why use a prompt chain workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.1.1: Prompt chains improve quality

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.1.2: Prompt chains improve testability

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.2: Understanding and designing prompt chains

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.2.1: Creating blog content

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.3: Building a prompt chain with Semantic Kernel

9.3.1: Overview of the workflow

9.3.2: Finding research online with the search tool

9.3.3: Outlining the article content with a prompt

9.3.4: Researching individual sections

9.3.5: Generating the article content

9.3.6: Finishing the workflow

9.4: Testing approach for prompt chains

9.4.1: Using property-based tests over model-based tests

9.4.2: Following the prompt chain with your tests

9.4.3: User testing

9.5: Optimizations of the prompt chain workflow

9.5.1: Adding auto-corrective steps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.5.2: Adding fan-out operations to parallelize the workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.5.3: Using intelligent routing to speed up the workflow even more

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

9.6: Summary

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10: Intelligent Request Routing Workflows

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.1: Why use intelligent routing in a workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.2: Introducing to the process framework

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.2.1: Installing the process framework

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.2.2: Writing the process steps

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.2.3: Wiring up the process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.2.4: Visualizing the process using mermaid diagrams

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.3: Making decisions in a Semantic Kernel process

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.3.1: Using events to route data through the workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.4: Building an intelligent request routing workflow

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.4.1: Configuring multiple AI connectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

10.4.2: Creating steps for straightforward and complex prompts

10.4.3: Routing prompts based on their complexity

10.5: Considerations for using intelligent request routing

10.6: Summary

11: Working with agents

11.1: What is and isn't an agent

11.1.1: The product marketing definition of an agent

11.1.2: The scientific definition of an agent

11.1.3: The role of large language models in agents

11.1.4: The structure of an LLM-based agent

11.1.5: The role of instructions in an LLM-based agent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.1.6: Which model to use for building agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.2: When to use an agent

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.3: Agent use cases

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.4: Building an agent with Semantic Kernel

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.4.1: Setting up an agent project

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.4.2: Creating an agent class

11.4.3: Invoking the agent with a description of a feature

11.4.4: Connecting the agent to other content in your project

11.4.5: Getting structured output from the agent

11.5: Building multi-agent systems with Semantic Kernel

11.5.1: Multi-agent patterns

11.6: Testing agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.6.1: Improving agents with tracing

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.7: Security practices when working with agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.7.1: Limit access to your systems

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.7.2: Be aware of data poisoning

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.7.3: Limit the autonomy of the agents

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/effective-llm-applications-with-semantic-kernel.

11.8: Summary