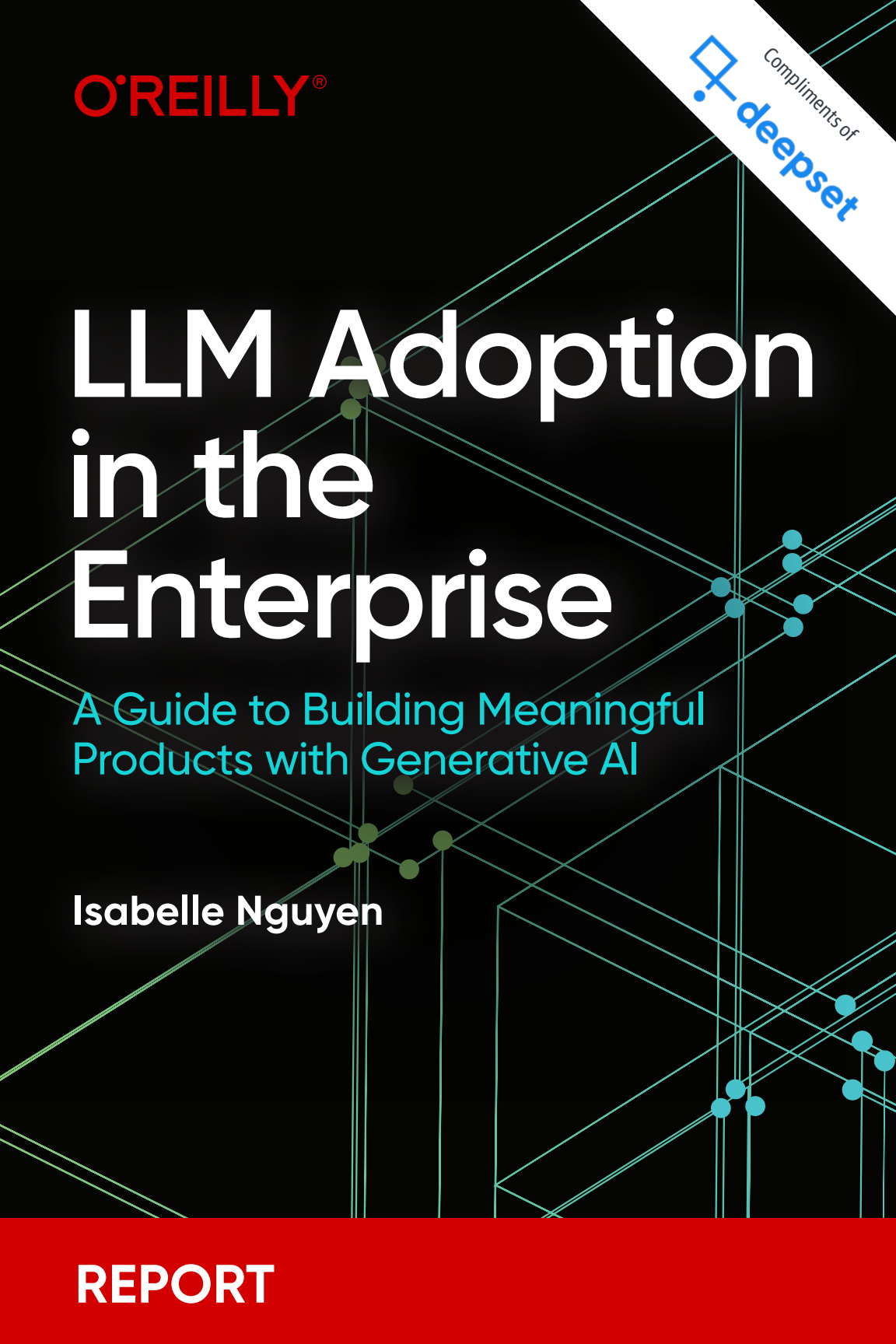
O'REILLY®
Compliments of
deepset
LLM Adoption in the Enterprise
A Guide to Building Meaningful Products with Generative AI
Isabelle Nguyen
REPORT

deepset

Fast-track AI innovation with **deepset Cloud**

Bypass the intensive DIY development process and speed time-to-launch and time-to-value.

Test 100+ architectures

In less than one month, iterate rapidly through pipelines with efficient prototyping

Index 10M+ documents

Move quickly and accurately to production by leveraging your complete dataset

Launch quality applications

High performance and high accuracy

FULL DIY VS. BUILDING YOUR SOLUTION ON DEEPSET CLOUD

	PEOPLE	COMPLETE PILOT	PRODUCTION	INFRASTRUCTURE	EFFICIENCY
Homegrown tech stack + components	8 FTEs	6 MONTHS TO IMPLEMENT	12+ MONTHS	HAS TO BE BUILT AND ORCHESTRATED	DIVERSE TEAMS MUST SELF-ORGANIZE AND TACKLE STEEP LEARNING CURVE
deepset Cloud toolchain + infrastructure	2 FTEs	1 MONTH TO IMPLEMENT	WEEKS	SCALABLE & RELIABLE INFRASTRUCTURE	END TO END LIFECYCLE SUPPORT and expert guidance from the deepset team

Discover more at deepset.ai/deepset-cloud

LLM Adoption in the Enterprise

A Guide to Building Meaningful Products with Generative AI

Isabelle Nguyen

Beijing • Boston • Farnham • Sebastopol • Tokyo O'REILLY®

LLM Adoption in the Enterprise
by Isabelle Nguyen

Printed in the United States of America.

Published by O'Reilly Media, Inc., 1005 Gravenstein Highway North, Sebastopol, CA 95472.

O'Reilly books may be purchased for educational, business, or sales promotional use. Online editions are also available for most titles (*http://oreilly.com*). For more information, contact our corporate/institutional sales department: 800-998-9938 or *corporate@oreilly.com*.

Acquisitions Editor: Nicole Butterfield
Development Editor: Gary O'Brien
Production Editor: Beth Kelly
Copyeditor: nSight, Inc.
Proofreader: Helena Stirling
Interior Designer: David Futato
Cover Designer: Karen Montgomery
Illustrator: Kate Dullea

April 2024: First Edition

Revision History for the First Edition
2024-03-27: First Release

978-1-098-15162-1

[LSI]

Table of Contents

Introduction vii

1. AI in the Enterprise 1
AI in Context 1
So What Exactly Is an LLM? 3
A Snapshot of Language Models in the Enterprise 4
Four Sample LLM Applications 5
Pivoting Attention from Technology to Product 8

2. Building an AI Product 9
What Is an AI Product? 9
Considerations for Building with AI 10
From Product Ideation Toward Planning 14

3. The AI Product Development Lifecycle 15
An Exemplary Case Study: Building a News Digest App 16
Developing a Product Hypothesis 16
Prototyping, Experimentation, and Evaluation 18
Refinement 19

4. The AI Team's Toolkit 21
Model Selection 22
Prompt Engineering 23
The Role of Data 24
Advanced Composable LLM Setups 27
Putting the Pieces Together 30

5. From Pilot to Product. 31
Scalable Pilots 31
Getting Ready for Production 32
The Challenge of Monitoring LLMs 32
Keep an Eye on AI Regulation 33

Building Meaningful AI Products. 35

Introduction

Tailored copy written by a chatbot, automated code reviews, and a fully AI-driven yet empathetic and helpful customer support. The potential of large language models (LLMs) to increase productivity and drive return on investment (ROI) seems too good to be true. But this disruptive technology also presents a challenge that goes beyond cost and technical know-how. To bring an LLM-based application to production, product leaders need a pragmatic understanding of the technology, the business outcomes it can drive, and the skills required to bring their product idea to life. They must be able to lead diverse teams through the adoption process and respond to new challenges as they arise. By learning to overcome their initial awe of a new and groundbreaking technology, product leads can begin to embrace the potential of LLMs to address their users' pain points. In this way, LLMs become a powerful addition to the product team's toolbox that, when used properly, can solve a variety of text-based problems.

The need for such a mindset shift is underscored by a recent Gartner® report,[1] which advises decision makers to learn "how generative AI can drive strategic value." Otherwise, teams risk building ultimately useless generative AI products whose only purpose is to "demonstrate that it is possible to build something with generative AI, leading to only incremental improvements and ignoring the transformative potential of this technology. *This is a mistake.*" This report is here so that you don't make the same mistake.

1 Leinar Ramos, et al., "How to Pilot Generative AI", Gartner, (July 2023) (GARTNER is a registered trademark and service mark of Gartner, Inc. and/or its affiliates in the U.S. and internationally and is used herein with permission. All rights reserved.)

In this report, I want to address the barriers that prevent thought leaders from adopting AI. Over the next five chapters, a new picture of AI in products will emerge that will help you understand how this technology can pave the way for previously unthinkable innovation. After dispelling some common misconceptions about the technology itself in accessible language, I'll introduce the most common applications of AI in the enterprise in Chapter 1. Chapter 2 takes a closer look at AI in a business context, specifically the product mindset that leaders need to adopt if they want to successfully deploy an AI-powered product in production. I will then go through the entire development cycle of an AI product and illustrate it with a use case that takes us through the different phases in Chapter 3. After that, the report gets a bit more technical, with an introduction to the AI team's toolkit in Chapter 4, and finally an overview of how to transition from piloting to production (Chapter 5).

Hyping a new technology is easy if you don't have to show proof that it actually works in a business environment. As a reader, it can be difficult to cut through the noise and find concrete examples that illustrate the use of AI in the wild. To provide a more practical experience for my own readers, I have made sure to include many different examples of what actual LLM products can realistically look like.

CHAPTER 1

AI in the Enterprise

This chapter introduces the intuition behind LLMs, as well as the key concepts from the world of AI and machine learning that are necessary to understand LLMs. Equipped with these concepts, we'll move on to see the most common uses of LLMs in the enterprise today, before taking a deeper dive into four exemplary projects that encourage a pragmatic understanding of how these language models can be used in industry applications.

AI in Context

As a field that is rapidly evolving and constantly drawing new people into the conversation, many terms in AI (such as "artificial intelligence" itself) aren't very well defined. Let's clarify our understanding of them for the purposes of this report.

AI Today Is Mostly Machine Learning

Our everyday use of the word "AI" describes machines that mimic intelligent human behavior. The means by which they do this are not limited to any one technique, and of course intelligence itself is a fuzzy concept. Machine learning (ML), on the other hand, is the study of algorithms that use data to construct models (i.e., predictive representations) of a given domain.

Not all AI is machine learning. But the vast majority of AI technologies making waves today—such as generative AI for images or text—rely on machine learning as their driving force.

An Algorithm Trains a Model with Data

The concepts of algorithms, models, and data are critical in ML. However, their relationship—as well as the nature of an ML model itself—is often poorly understood. It is important to note that a model is not an algorithm, but rather *the result* of data being processed by an algorithm. An algorithm can be thought of as a plan or set of rules in a programming language that describes this process, which is also known as "training a model on a dataset."

The algorithm itself only defines the structure of the model, not its content. That's why running an ML algorithm on two different datasets—say, one in English and one in Spanish—will produce two different models.

A Trained Model Is Defined by Its Parameters

A machine learning model's size is determined by its parameters: adjustable numeric values that are optimized during training to accurately recognize and predict patterns in data. The values of a trained model's parameters characterize it and can be shared and reused, especially for inference, where the model applies its learning to new, previously unseen data points.

Large language models can have millions, billions, or even more trainable parameters.

There Are Many Different Kinds of Language Models

Language models have learned a representation of the language on which they've been trained. This representation can then be used to make predictions, such as which word will follow a sequence of words.

But not all language models are generative. In particular, smaller, pre-ChatGPT models that are still widely used in many different applications specialize in extracting information from text or embedding text in numerical format for fast semantic search. These more specialized models, such as RoBERTa and Sentence-Transformers, continue to exist alongside their generative LLM cousins, whose emergent properties have made them world famous.

So What Exactly Is an LLM?

Now that we've gone over these complex concepts related to machine learning, language modeling, and AI, we have the understanding and the terminology to talk about what LLMs are (and aren't).

In the most common understanding of the term, LLMs are language models with billions of parameters that excel at generating text in natural language. To ensure that they learn to capture an accurate representation of the world, LLMs are trained on vast collections of textual data. In addition, it is becoming increasingly common to find multimodal models that can process and generate both text and images—for example, they can be used to create captions for diagrams. This is a natural consequence of the fact that the technology behind LLMs is also very good at processing visual input. The technology underlying LLMs also excels at processing visual input, so they can be successfully applied to text and images simultaneously, as seen in models like GPT-4 and Gemini. By looking at terabytes of data from the web and other sources, LLMs use their many parameters to learn linguistic patterns, as well as factual information and correlative relationships across a wide range of topics. They can then use these representations to perform complex tasks, like writing valid code in any programming language, generating image captions, and composing prose in the style of a particular author from scratch. These abilities—known as "emergent properties," because they were not literally present in the training data—arise from the LLM's ability to apply the patterns it has learned to new domains. For many people, they are the most impressive feature of these language models.

An important concept in the context of LLMs is that of prompts. Prompts are the text that we, as users of an LLM, feed to the model, instructing it to perform certain tasks. As such, prompts are the de facto user interface of an LLM. Effective prompting requires some practice and the use of tested and proven techniques. The length of the prompt is model-specific, but most prompts have enough space to provide context for the model to base its responses on—a critical concept in the most successful enterprise applications of LLMs, as we will soon see.

What makes LLMs so attractive to businesses is their ability to reason and write in humanlike quality and at a scale well beyond that. Text-related tasks such as programming, report analysis, information extraction, and copywriting can now be performed by machines within seconds, opening up a wide range of applications previously unimaginable. Maybe that's why it can be hard to ideate use cases: we have to change our entire way of thinking about what's possible. To help us broaden our horizons and start thinking about LLM-powered applications that create value in the real world, let's look at practical examples of the most common uses.

A Snapshot of Language Models in the Enterprise

In practice, the boundaries between generative LLMs and their smaller counterparts are not always clear. All language models originate from the discipline of natural language processing (NLP), which has long been concerned with how to make natural, human language processable by machines. After early successes such as machine translation, spam detection, and speech generation, NLP finally entered the mainstream with generative AI. But now we're seeing a trend among industry thought leaders and analysts to recognize the usefulness of earlier language models by including them under the umbrella of LLM. These smaller, nongenerative models have had more time to gain traction and are already firmly and quietly embedded in products across industries—increasingly in combination with generative LLMs. In addition, a new class of small language models (SLMs) has recently emerged that aims to capture the generative and reasoning capabilities of LLMs, albeit with a much smaller number of parameters. Now that we have a rough overview of the language modeling landscape, here's a list of the four most important use cases for language models in production:

Generative AI and chat

Given the popularity of ChatGPT, it is no surprise that generative AI is the most popular use case for enterprises. In the context of LLMs, generative AI describes the generation of text in response to a prompt. For example, it can provide users of a styling service with customized outfit descriptions on the fly.

Information extraction
: Information extraction uses language models to isolate key pieces of information from text documents for use in downstream tasks, such as populating a table or database. Consider a construction company that works with tens of thousands of contractors, all of whom have different insurance policies. To ensure that it is not liable for its contractors' mistakes, the company wants to understand which risks are covered by which policies. To do this, it automatically extracts this information from the lengthy (potentially 500-page) contracts and stores it as metadata.

Recommendation and search systems
: Semantic recommendation systems take advantage of the ability of language models to mathematically represent and compare the meaning of text, providing an extremely fast way to find similar documents, even across large databases. An example of this technique in action is recommending properties with similar descriptions after each listing on a real estate platform.

Text classification
: Classification is a "classic" NLP application that can help sort text into predefined categories for faster processing. For example, you may want to classify incoming user queries as positive or negative before passing them to a downstream application or human agent.

Four Sample LLM Applications

The products best suited for LLM adoption have one aspect in common: textual data is a major factor. Here I look at four examples, inspired by real-world LLM applications, that serve to further illustrate how companies can move from problem to LLM solution. You'll see that they span different industries, pain points, flavors of LLM, and user groups.

Sophisticated Recommendations for Legal Professionals

For lawyers working on a case, it's critical to review all relevant legal cases and documents. Consider a company that has a large database of legal texts—including past cases and decisions—available to its

subscribers, most of whom are legal professionals. The company wants to enhance its service by adding a feature that displays a list of related cases and documents on its platform. This tool is designed to help lawyers find and review relevant cases more quickly and effectively, thereby allowing them to take on more cases or improve the quality of their work by catching details they might otherwise miss. However, because new cases are constantly being added to the database, this list of relevant cases cannot be static, but must be defined dynamically.

The solution is a recommendation system based on semantic similarity. Semantic similarity uses small or large language models to determine the closeness of meaning between two documents based on an abstract representation of their content, rather than the vocabulary they use. This helps to identify two semantically similar documents even if they use very different words. In addition to the search functionality and its integration into a user-facing frontend, the team tasked with building the application also needs to architect and schedule the preprocessing of incoming documents. This process, known as indexing, is accomplished by the same language model.

Conversational AI for Technical Documentation

For developers, clear and accessible software documentation is critical—it is the guide that helps them use software or libraries effectively. But navigating documentation can be daunting. Consider the massive scale of cloud service providers like Amazon Web Services (AWS) or Azure: their documentation can run to thousands of pages, covering a wide range of services, features, and policies. Faced with such a sea of information, users quickly feel overwhelmed and often turn to browsing the web for answers rather than sifting through official documentation. To address this problem, a software company wants to revolutionize the way customers interact with their documentation. They plan to introduce an intuitive search interface that allows users to ask questions about the codebase in natural language and quickly find accurate answers. This not only streamlines the process, but also ensures that developers are getting the most accurate and relevant information directly from the source.

The natural language capabilities of an LLM are ideal for this use case. However, the team doesn't want the LLM to generate answers

based on the knowledge it learned during training: the documentation was probably not part of that training data—and even if it was, that information could become outdated with the next software update. To compound the problem, LLMs are notoriously bad at understanding the limits of their own knowledge. Thus, when asked about things absent from their training data, they often answer regardless—by inventing facts. This is known as "hallucinating," and it is a major problem in LLM adoption.

The remedy to outdated information in the LLM's parameters and ensuing inflation of hallucinations is a method known as retrieval-augmented generation (RAG). In such a setup, the LLM is preceded by a retrieval module, which extracts the candidate documents it deems most suitable to answer the user query from your database. Upon receiving a query, the RAG system first identifies suitable documentation pages. It then embeds those in the prompt to the LLM, instructing it to base its answers on the fact-checked information from the database. RAG as an LLM technique is extremely popular because of its ability to create a factual knowledge base on the fly.

Automating the Collection of Information from Earnings Reports

The advent of LLMs has enabled machines to process unstructured data. The term "unstructured" refers to data types such as images, audio, video, or text: formats that don't follow a strictly predefined structure. Structured data, on the other hand, is data that comes in a predictable format, such as tables and graphs, and can be processed using less resource-intensive methods. For example, a large table can be queried using SQL, which is faster, more accurate, and infinitely cheaper than running an LLM for the same task.

Let's say a company wants to identify information in its unstructured textual data that can be fed into such tables. For example, they might want to extract specific numerical and other factual data points from a collection of earnings reports.

The solution: using a smaller language model to mine text for information, not by generating answers, but by highlighting it in the underlying source document. Such models are called "extractive" language models. They're not only lighter and cheaper than LLMs but also safer for highly sensitive areas such as finance. That's

because they are incapable of LLM-like hallucinations, and they are necessarily more faithful to the underlying dataset.

Condensing Political Discourse for News Consumers

Democracy thrives on the active participation of its citizens. But political debates in parliament are often long and difficult to access. For this use case, let's imagine a government application that wants to make parliamentary debates more accessible to citizens by summarizing the debates' transcripts according to the user's interest.

This use case is similar to the second in that we're dealing with a database of texts that is updated periodically. And indeed, to address it, we would again use a retrieval module that extracts the relevant transcripts upon receiving a user query. But instead of generating answers, this system would use an LLM to summarize all the underlying texts. In this way, users could get timely overviews of political debates tailored to their individual interests.

Pivoting Attention from Technology to Product

New technologies are often the subject of hype at one end of the spectrum and doom at the other. The same is true of large language models, whose complexity makes them difficult to grasp even for technically-minded people. But to build useful and innovative products with AI, you don't really need to understand the mathematics of LLMs or the details of their implementation. What you do need is a strong sense of how they can be applied to solve real-world problems. This chapter has introduced basic concepts in AI, with a specific focus on LLMs and how they can be used. The practical examples of AI-powered products in the wild have given us a sense of how we need to start approaching LLMs to be successful in building with them. In the next chapter, I'll examine the notion of an AI product and what you need to consider before you start building one.

CHAPTER 2

Building an AI Product

As the sample projects in the previous chapter have shown, gaining a solid understanding of the technology is just the beginning of building with LLMs. This is true for any technology, but because of the complex and intimidating nature of these recently introduced models, a closer look in the context of AI in particular is warranted. So in this chapter, I want to explore the notion of an AI product and how you need to start thinking about the technology to build products that will drive your business forward.

What Is an AI Product?

An AI product is an application, a piece of software, or an actual physical product that uses the capabilities of one or more AI technologies to process data and deliver a specific user experience, coupled with a business outcome. When we talk about building with LLMs and AI in general, AI itself is not the product. Rather, it is the technology—often, the central technology—that powers the product. For example, the ChatGPT browser application is powered by an LLM. But it also consists of many other features, such as the chat interface, user verification, and some sort of memory component to store the previous conversation.

Product ownership is about understanding the process of developing a product from start to finish, and not losing sight of the end goal. To do this, it's useful to adopt a "product mindset"—that is, a methodical approach to the project as a whole, as well as to its individual parts. In that mindset, LLMs are an exceptionally powerful

tool for tackling language-based tasks. And this doesn't only apply to language: adopting a pragmatic view of generative AI technology in general can unlock its untapped potential for a wide range of products.

LLMs can help accelerate processes across the enterprise, with potential user groups inside and outside your organization. While there are obviously different considerations around requirements, UI design, communication, scale, and security, there is no technological difference between internal and external products. In fact, a 2022 Forrester survey found an almost even split between AI-powered projects to automate processes within an organization (39%) and those deployed in customer-facing products (33%).[1] So when we talk about implementing AI in the enterprise, we should always consider both.

Considerations for Building with AI

When developing an AI product, there are several aspects to consider, including the business purpose, the technological implementation, and the skills required. Identifying the business purpose is a critical and fundamental step in product management, irrespective of whether it involves AI or not, as it helps you set the boundaries for your project. It's crucial to ask the right questions at this stage. Specifically, what is the problem you want to solve? Who are the end users who will benefit from the solution? Which subset of their problems does the solution address—and which does it not? What new capabilities does it create? In short, think in terms of use cases rather than technologies.

Defining a Use Case

The term "use case" has slightly different meanings in technical and business contexts. In a technical context, it refers to a description of how a user interacts with a system, detailing success and failure scenarios and highlighting notable edge cases. In a business context, a use case serves a similar, albeit more comprehensive, purpose. Often narrative in form, a business use case explains the rationale behind a product and aligns it with the company's overall goals.

1 Forrester Research, "Artificial Intelligence Market Insights, 2023", May 2023. Available to Forrester subscribers or for purchase.

It simplifies complex concepts, and is intended to be understood by all stakeholders, not just those with technical expertise. A well-constructed business use case provides decision makers with critical information about market opportunities and the product roadmap. It presents a clear and compelling vision of the product throughout its lifecycle, helping to assess and mitigate risk.

In the product development lifecycle, a business use case always begins with the identification of a problem to be solved. Conducting market research and user interviews is critical to gain a deeper understanding of the problem and identify the ideal solution, ensuring that the product we market solves a significant, real problem. The challenge that this step poses should not be underestimated: "Difficulty finding appropriate use cases is the biggest bar to adoption" in enterprises.[2] At the same time, it's the key to making sure that products become a success. A use case is therefore more than just a description of a user–system interaction; it's a comprehensive tool that guides the entire product development process, from ideation to launch, ensuring that the product is not only technically feasible, but also aligned with business goals and user needs.

Understanding the Technology

Your use case drives the technology you build your product with, not the other way around. The biggest piece of advice here is not to be intimidated by AI and the noise surrounding it. The internet as we know it today rewards the loudest voices with the biggest claims, and we've seen a lot of unfounded predictions before an AI product has even shipped. Trust your instincts—if people are promising products that seem to be powered by magic rather than the LLM techniques we know, they're probably more fantasy than reality. If you're unsure about what this technology can and cannot do, reread the "AI in Context" section in Chapter 1 of this report and research any concept you don't fully understand. Find trusted experts inside and outside your organization to talk to. Although engineers who are good at explaining technology to laypeople are a rarity, you can take advantage of the many blog posts out there whose job it is to do just that. Try to find case studies of actual AI products that have been deployed in production. Reading and hearing about what other

2 Mike Loukides, "Generative AI in the Enterprise," O'Reilly (November 2023).

people are building (and the problems they're running into) can give you a good sense of what's actually possible with AI. What's more, understanding what LLMs are really capable of will help you spot empty promises from AI vendors—and help you know when a new development is really worth getting excited about.

Beyond the technology itself and its application to different business use cases, the most important considerations when it comes to LLMs may be related to cost and privacy. Sometimes there is a trade-off between the two. Therefore, your requirements for one of these factors will often influence how much leverage you can apply to the other. For many organizations, the first instinct is often to "*just run an open source model locally for maximum security*," not realizing that such a solution would be prohibitively expensive to build and maintain. But even deploying a freely available LLM on a cloud platform like AWS or Azure can be more expensive than using a proprietary LLM via the vendor's API, thanks to the latter's on-demand business model. When exploring such proprietary options, it's useful to understand the privacy restrictions and requirements associated with different jurisdictions: the same model running on servers in different locations may process and store your data differently. Understanding the cost–privacy trade-off and how it relates to your use case is a useful exercise that takes some practice.

Assembling the Right Team

Building an AI product requires a variety of skills, from data science to software development to user experience (UX) design. Cross-functional teams are important for the success of an AI product development project. Because of the inherently interdisciplinary nature of AI and its applications, its technical complexity and cost, and its potential business impact, you need people on your team who can give the project the many points of view it deserves. Here, I want to focus on three roles that deserve special attention in the context of building and delivering a product: AI engineers, because the function of this emerging field is sometimes poorly understood; subject matter experts, because their unique insights are often neglected; and business representatives, because without their enthusiastic support, your product won't get the funding it needs to get off the ground.

AI engineers

Data scientists and ML engineers have a very specific skill set that revolves around data and models. Traditionally, they spend a lot of time curating datasets and training, evaluating, and improving machine learning models. However, these skills are less relevant when it comes to integrating LLM technology into a product. That's why a new role has emerged: the AI engineer. AI engineers are pragmatic builders with a product mindset. Rather than focusing on curating data to build models that will later become products, they think about how AI can enrich products and what data, if any, they need to do so. AI engineers understand the LLM space and aren't intimidated by it. Among their many in-demand skills—which we'll explore in more detail in Chapter 4—is expertise in prompt engineering. The current moment in LLM development is encouraging pragmatic-minded AI enthusiasts from diverse backgrounds to reinvent themselves as AI engineers, united by their passion for turning AI technologies "into real products used by millions, virtually overnight."[3]

Subject matter experts

Subject matter experts (SMEs) are typically much less technical than the rest of the team. Their value comes from their industry-specific knowledge and deep understanding of the use case. In the context of our semantic recommendation system for lawyers, these would be legal professionals, whereas a healthcare application would require the expertise of doctors, nurses, and other medical personnel. As Chip Huyen writes, "SMEs (doctors, lawyers, bankers, farmers, stylists, etc.) are often overlooked in the design of ML systems, but many ML systems wouldn't work without subject matter expertise. They're not only users but also developers of ML systems."[4] The same is true for AI products, where SMEs play a critical role in shaping the use case, curating and annotating datasets, and testing the product through its various iterations.

3 Shawn Wang, "The Rise of the AI Engineer", Latent Space (June 30, 2023).

4 Chip Huyen, *Designing Machine Learning Systems*, O'Reilly (2022).

Business representatives

Engaging the business early in the development process is essential to ensuring that the product aligns with the company's strategic goals. What's more, the active support of C-suite executives and business representatives secures the financial and organizational backing of the project. Without this corporate commitment, even the most visionary project risks stalling because it lacks the resources to get off the ground and scale. By involving the business from the start, you avoid potential disconnects between the technological capabilities of AI and the practical needs or constraints of the business. You'll ensure that every step—from ideation to development to deployment—is taken with a clear understanding of the business context, ensuring that the end product is not just technologically impressive but also a strategic asset.

Cross-functional teams are emblematic of a holistic approach to product development, encouraging the collaboration and innovative problem-solving that's critical to overcoming the challenges inherent in AI projects. The diverse insights and expertise within these teams ensure that the AI solution is not developed in a vacuum, but is a product of collective intelligence, closely tied to real-world applications and business goals.

From Product Ideation Toward Planning

Once your team has identified one or more viable use cases for incorporating LLMs into the company's products, it's time to start planning the adoption process. The dominant paradigm in software development today is an agile workflow consisting of rapid, iterative cycles. Building with AI is no different. In fact, I'd argue that it's even more important with a technology as unexplored and expensive as AI, because frequent reality checks with your team and potential users will ensure that your organization's resources are well spent and that you're building a product with real value. In the next chapter, I will discuss the development cycle for building AI products, using realistic examples to guide us through the process.

CHAPTER 3

The AI Product Development Lifecycle

Software products benefit from short development cycles coupled with frequent feedback. The same is true for building with AI. Rapid prototyping, user experimentation, the data and security constraints you may face, and the volatile nature of LLMs in particular make experimentation and reorientation of your product a feature rather than a bug of the AI product development cycle, illustrated in Figure 3-1.

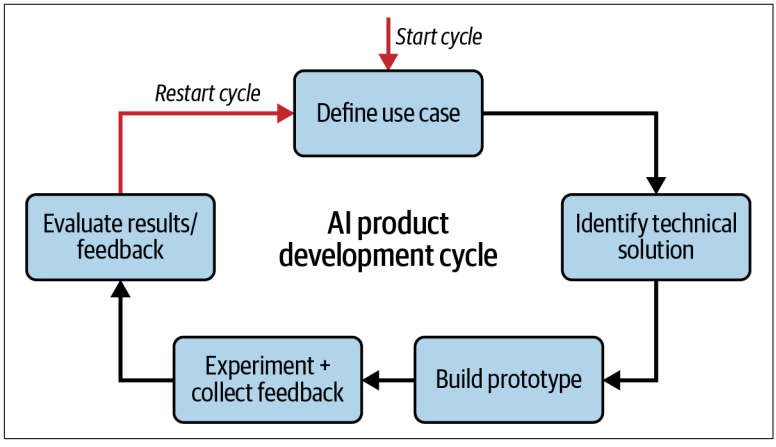

Figure 3-1. A sketch of the development cycle for AI products; the cycle can be repeated hundreds of times before the AI team arrives at a viable product

To ensure that your project stays on track, this cycle consists of several iterations of defining the use case, building a solution, experimenting with it, and evaluating the results, after which the cycle begins again. To help us get comfortable with this way of working, in this chapter I want to take you on a guided tour of the steps in the AI development lifecycle.

An Exemplary Case Study: Building a News Digest App

To understand the AI product development lifecycle, let's consider the following hypothetical case. An online news platform is experiencing a dwindling number of subscribers. They know they're up against a powerful competitor: social media channels are increasingly taking over the role of traditional news media by being the first place people go to find out what's happening in the world. This is especially true among younger demographics, making it all the more urgent for news media publishers to explore new technologies and find new ways to engage with their audiences. Of course, everyone in the company has seen ChatGPT in action, and many understand that LLM technology can be a game changer for all industries, and especially for publishing. They want to use LLMs but are unsure about their capabilities and the skills required to operate them. There's also uncertainty about how a product with generative AI capabilities can retain and attract users.

Developing a Product Hypothesis

To start building something, you must first identify a problem. In the field of AI, executives are often in a rush to turn the technology into a product without first considering the actual problems that existing users face and that AI could potentially solve. But it's the product owner's job to identify user pain points and, from there, develop a hypothesis based on their needs and expectations. Ideas for products can come from past usage patterns or problems that customers have explicitly mentioned. Successful ideation requires alignment with people across your organization and with your users. Such a process takes time and is supported by a culture of careful listening to users, the market, and internal discussions. But insights can be gained through a variety of tools, such as user research and feedback from social channels.

Generating a hypothesis for your product can be like a litmus test for how well you understand your users. And even if your initial experiments prove you wrong, you and your organization have everything to gain: insights about your users, as well as potential ways to improve your product and company culture.

Defining the Problem

In our online news publisher example, a product team wants to build a compelling new feature into the product using generative AI. They want to build an initial product pilot as soon as possible to prove to management (and the organization as a whole) that investing time, effort, and resources in AI and LLMs is the future of publishing. The team doesn't have a specific product idea yet, but after seeing the impressive writing capabilities of OpenAI's models firsthand, they imagine that the organization could use them to generate new content on the fly, with an LLM customized to the voice of the platform.

To solidify the use case, the product lead conducts a half-day workshop with a cross-functional team consisting of stakeholders from business, engineering, data science, marketing, and the content team itself. During the workshop, however, a different sentiment emerges: users seem less interested in new content and more interested in consuming existing content. In fact, team members who are in close contact with the platform's users report a pervasive sense that a constant stream of events is making it harder to keep up with what's happening in the world.

Even though subscribers can personalize their news by listing the topics that interest them, they're often overwhelmed by the amount of content and feel they have less and less time to read all the articles that interest them. As the product lead discovers when she sends out an initial questionnaire to some of the power users who have agreed to be surveyed, they often end up reading articles that don't really align with their interests, while feeling guilty about not keeping up with more important news and wasting their subscription. Guilt is the last feeling you want your readers to associate with your product! The product team sees an opportunity for a new feature powered by LLMs.

Identifying a Technical Solution

During the workshop, the cross-functional team agreed on a preliminary use case: to build a feature into the platform that summarizes the latest content based on the user's interests, which they select manually from a list of topics. To do this, the team wants to build a RAG pipeline that retrieves articles from the database and summarizes them. The articles are selected by topic and publication date. In their prompt to the pipeline, the AI engineers instruct the LLM to create a summary of the articles it receives.

Prototyping, Experimentation, and Evaluation

Prototyping allows teams to learn how users react to the product idea in a live environment. AI teams should embrace prototyping and experimentation because it allows them to test whether their initial hypothesis holds up. After deploying your prototype, it's a good idea to share it as early as possible with a select audience through an easy-to-use UI. Test users should be able to stress test the feature by running it with multiple queries or different configurations so that they can provide comprehensive feedback on the functionality of the feature. This is an exciting milestone in any product journey, as people begin to apply your prototype to their real-world problems. Prototyping also gives you the opportunity to discover additional pain points and unexpected usage patterns.

What you are testing and refining at this stage is not the LLM, and not even just your particular application of an LLM to a use case. Rather, it is the product as a whole. For example, you may find that your application requires specific adjustments to the UI. Or you may find that users' expectations of the documents in your underlying database differ from what your product actually uses.

Shipping a Prototype

Going back to our news digest feature, the engineers on the team built a prototype interface for the first test users. The feature is resonating with these early testers. They like the idea of getting readers up to speed quickly and giving them a starting point from which to explore the news. After an initial round of positive feedback, the team runs a round of A/B testing with the new feature. While testing with 200 additional users directly in the app, they notice a curious

pattern. After scrolling through the summaries, many users turn to the app's search function to verify that what they just read is true. It seems that people do not easily trust the new feature.

Refinement

This is where our AI product development cycle begins anew. By observing that some test users are not sure whether the new, LLM-powered feature is fact-based or hallucinating, our team has identified a new pain point and must look for a new solution to address it. However, it's important to keep the big picture in mind during this phase. Changing one aspect of the new feature might make another aspect worse.

Back to the Drawing Board

The LLM experts on the team have worked hard to perfect the RAG setup and prompt, and internal evaluations have shown that the summaries generated by the application are factual and don't contain hallucinations. However, this is not obvious to users who try to verify the claims themselves. To address the need for users to research the events mentioned in the generated summaries, the team comes up with a straightforward solution: they tweak their prompt to include citations after each statement. The citations refer to the retrieved articles, which are then listed at the end of the summary so that users can go to them to learn more. In this way, readers can fact-check the information referenced in the automated summaries or simply read more about topics that interest them.

Another Round of Experiments, Refinement, and… More Experiments

In the next round of experiments, test users praise the addition of citations and references in the new summary feature. However, these new experiments also show that the summaries are not as good as before. This is probably due to the fact that LLM engineers had to make extensive changes to the prompt used to generate the citations. Prompting is a delicate art, and changes to a prompt in one aspect often result in undesired behavior in another. That's why it's important to keep a checklist of the growing number of things your prompt must accomplish. It's also a reminder that when we build with LLMs, we need to remain vigilant about the model's

behavior over time and monitor its performance in production. Now that we've discussed the workflow of an AI project, in the next chapter I'll discuss the tools and skills needed to build effectively with LLMs.

CHAPTER 4
The AI Team's Toolkit

So far, we've focused on the soft skills your adoption process will require, such as intuition, observation, and effective cross-functional communication. But your AI adoption cannot be successful without hard skills as well. Let's say you've assembled the kind of cross-functional team described in the previous chapter that's needed to get your product off the ground. Each role has its own unique set of skills. When we focus on AI teams, especially AI engineers, we're considering a wide range of interests, talents, and tools. In this chapter, I want to explore the key components of this diverse skill set.

In addition to keeping up with the latest developments in language modeling, AI engineers are often very good prompters, thanks to their hands-on experience getting LLMs to do their bidding. Because effective prompting is so important to getting the results you want, avoiding hallucinations, and even saving money, I will look at it from a slightly different perspective in this chapter: *prompting as programming in natural language*. Then, we'll move on to the most valuable resource of our time—data—and best practices for AI teams to take care of their organization's data assets. We'll see why well-governed and curated data is a necessity for the vast majority of generative AI applications in the enterprise. Finally, I'll talk about how the composability principle of modern-day natural language processing (NLP) designs allows AI engineers to create ever more complex and powerful LLM applications.

Model Selection

For someone just starting their generative AI journey, the sheer number of language models—built for different purposes, sizes, languages, and modalities—can be overwhelming. But staying on top of what is possible with LLMs is one of the most important practices of an AI engineer. How do they do that? Often, those seeking information about the best, fastest, or otherwise high-performing models are directed to leaderboards and benchmarks. Leaderboards run a series of standardized tests on LLMs and then use the results to rank the models from best to worst. They are useful for understanding how well a new LLM performs against established models. They also give you a good sense of what metrics the broader AI engineering community uses to understand the quality of a model's performance. Another popular and entirely community-driven format is the "Chatbot Arena Leaderboard" hosted by LMSYS, which uses an Elo rating system (like chess) to determine which chatbots users prefer.

But while these can be useful tools for getting an overview of what's out there, they're more oriented toward research rather than business. Unlike academic research, business use cases must cater to many different requirements from various stakeholders. When deciding which LLMs to use in your product, it's therefore important to ask the right questions:

- Do your business requirements dictate that you work with an open source model, or can you use one of the proprietary LLM APIs?
- How important are issues like latency (that is, the time between sending the request and receiving an answer) to your users?
- Does the model need to cater to specific languages or other skills, such as programming languages?
- Would your application benefit from a larger context window?

To narrow down the range of models, it makes sense to talk to the AI engineers in your circle who know the most about these issues. They, in turn, often keep up with model developments through social media and by following thought leaders in the space. Note that, as we've highlighted in the previous chapter about the iterative process of AI product development, you do not need to set your

model of choice in stone at any point in your product journey. In fact, it often makes sense to start with the most powerful, state-of-the-art LLM and experiment with it to understand what's possible. Then, you can plug in smaller models associated with lower cost and latency and see if you can use prompting and other techniques to get them to perform at a level similar to the larger model.

Prompt Engineering

As we developed our hypothetical use case of a news digest app in the previous chapter, we've already got a glimpse of the role of the prompt in the context of LLMs. While AI practitioners have explored the capabilities of prompting, it's become increasingly clear that with LLMs, the prompt acts as a programming interface for users. More than just the query input interface, prompts can be used to transmit data to the LLM, as well as guidance through examples in what is known as "few-shot prompting," and measures to prevent unwanted behavior, like hallucinations. To use an LLM in an effective manner, AI teams need to understand that the right prompt can greatly affect the model's performance. The emergence of structured prompting languages such as ChatML exemplifies the move toward more formalized and efficient interaction with LLMs, resulting in a safer and more controlled conversation flow.

Mastering the art of prompting takes practice. While there are many more or less evidence-based tips on how to prompt effectively, the most consistent advice is to be overly explicit and literal about what is being asked of the model (since LLMs are bad at reading subtext), and to double down on any points that are particularly important. Subject matter experts (SMEs) are often good writers of prompts because they have a deep understanding of their domain. For niche topics, it often makes sense for AI engineers to team up with SMEs, since their expertise can lead to more precise and effective prompts.

Because different user inputs can elicit a wide variety of responses, it's important to use the prompt to control and direct the LLM's output. This includes defining the tone, length, and whether the response should include citations. These requirements will generally depend on the specific needs dictated by your use case. Once a prompt is used with an actual LLM, its responses will in turn lead to adjustments to the prompt.

Among AI engineers, prompting is seen as a fun and creative process, where they can use the power of words to unlock AI capabilities. However, small adjustments to the prompt can lead to disproportionate variations in the LLM's output: sometimes even a small change in punctuation can change the result significantly. A curious and experimental attitude is therefore crucial to prompt engineering.

It's also useful to test prompts with real users. This can help determine not only how well a prompt works, but also whether it's ready for real-world use. After all, a prompt can always be tweaked and improved, so user feedback is an essential part of determining when it's really ready. As we'll see later in this chapter, documenting your work is crucial in any AI project. The same holds true for prompts: since their shape and wording have such a high impact on the LLM's output, it's a good idea to track all changes to the prompt and how they affect the results.

The Role of Data

Without data, ML/AI as we know it today wouldn't exist. You've probably heard some variation of this sentiment a million times. It's clear that large and high-quality datasets are essential for training ML and AI, but what is their role in product development?

More than just a necessary resource to feed LLMs, enterprise-specific data can actually be what differentiates your product from the competition. Think about it: across your industry and beyond, all AI teams have equal access to the same LLMs—but for a particular company, it's the proprietary data, collected over the organization's lifetime, that makes it unique and sets it apart. With that in mind, let's look at the three main ways that data can enrich, shape, and characterize LLM-based applications.

Training and Fine-Tuning

Ninety-nine percent of organizations will never train an LLM from scratch. Not only is LLM development prohibitively expensive, it's also unnecessary, given the wealth of models already available. There is, however, a technique for further adapting an existing LLM to a specific use case. *Fine-tuning* is the process of improving the parameters of a pretrained model to fit a particular use case. This use case is represented by the domain-specific data the model sees

during the fine-tuning process. For example, you might fine-tune an LLM used in customer service to match the voice of your brand better, using transcripts of previous real-world interactions with your clients. Or, if you're building a RAG setup for business intelligence tables, you might want to fine-tune the retrieval model with the schemas in your database.

Fine-tuning requires much less data and computing power than training a model from scratch, and it has been used successfully to adapt a model to a particular tone or task. As a technique for feeding information into the model, however, fine-tuning is much less effective than RAG. That's because it doesn't reliably remove hallucinations. Moreover, if your goal is to update the LLM's knowledge base, this means that you would have to fine-tune it for each new piece of data, which in turn will increase your costs significantly. However, for smaller language models, such as those used in retrieval, fine-tuning is often useful because it improves performance without consuming too many resources.

Evaluation

To understand how well your model or overall application is performing on your specific use case, it's necessary to perform periodic quantitative evaluations. To do this, you need annotated data that represents your use case as accurately as possible. This data is then run through the model and various metrics are used to quantify how well the model's predictions match your own annotations. Note that while quantitative evaluation of retrieval and classification tasks, for example, is well established, it's much harder to quantitatively evaluate LLMs in an equally satisfactory way—though I'll discuss some promising approaches in Chapter 5. It's therefore often complemented with user feedback—a much more high-quality and costly method for LLM evaluation that is based on real users rather than evaluation datasets.

Retrieval Augmentation

Large language models are powerful but slow and expensive compared with other models. Retrieval augmentation, which we already discussed several times in this report, uses fast data-matching techniques to preselect the most appropriate data from your database to pass on to the more resource-intensive LLM. The type of data is completely dependent on your application, and could be anything

from business intelligence tables to social media posts or comic books. What's clear is that, in a retrieval-augmented setup, the quality of the data directly affects the quality of the LLM's output. That's why it's important to have data curation practices in place that ensure the data is current and accurate.

Best Practices for Data Curation

In a survey sponsored by AWS among chief data officers, poor data quality was seen as an obstacle to AI adoption at the same scale as "finding a use case" (listed by almost 50% of respondents).[1] Keeping data relevant and of high quality is an ongoing task. Just as real-world products must adapt to changing user needs, data must evolve with them to ensure it still reflects your use case. Regular quality checks and updates ensure that an LLM in production continues to perform optimally as conditions change. Stress testing LLM applications with complex data points can also provide valuable insight into the resilience of these systems.

The annotation process is a cornerstone of data quality. Annotation (also known as labeling) describes the process of attaching labels to your data, which the model then has to predict. At its simplest, these labels or categories can be binary, such as in retrieval or a basic type of sentiment analysis. However, when dealing with natural language, annotations are often much more complex. For example, annotating response sequences for extractive language models requires labelers to determine the start and end token of a response.

Much like teaching people to drive, annotation is a repetitive task that still requires a lot of attention. As we've seen, the quality of your annotated data directly affects downstream tasks such as fine-tuning, retrieval augmentation, and evaluation. To ensure efficient workflows, it's important for organizations to establish clear and understandable guidelines that are tested on real data before they're put into practice. Peer reviews and cross annotations ensure that annotators are applying the guidelines to the data in a consistent manner.

1 AWS, "2024 CDO Insights: Data & Generative AI".

With the frequent iterations of AI product development, it is easy to lose track of what data was used in a given cycle. Thorough documentation is therefore essential—not only for code, but also for datasets. It helps keep track of important metadata, such as when and where a particular data point was created, as well as the context in which it was created. Depending on your use case, you might want to augment your internal company data with external datasets. AI teams need to make sure they understand that data well before using it to build any products, by using established data exploration techniques from data science. It's important to know the origin of the data, the annotators involved, and the conditions under which the annotations were made. This transparency is not only helpful for current use, but also for future reference and understanding.

Advanced Composable LLM Setups

LLMs are advanced pieces of software all on their own, but the composability principle in modern AI applications makes it so that we can, in theory, build increasingly complex and powerful architectures. For example, it allows us to use multiple language models in one product, or even embed different modalities like images, sound, and text. Here, I want to take a brief look at some advanced concepts that will likely gain traction in the following months.

What Is Composability?

Composability is a well-established concept in applied NLP. It allows AI engineers to build modular applications consisting of multiple parts that work independently but can be combined for a different experience. The classic RAG application uses this principle, for example, by stacking a generative LLM on top of a retrieval module, resulting in a product that is more powerful than the sum of its parts. In addition to this extended capability, composable systems have two major advantages:

- Parts of the architecture can be replaced independently. This is critical in a field that evolves as rapidly as LLM development.
- Models can be evaluated independently, making it easier to find sources of error in a complex composed system. This is especially important since some language models are easier to evaluate than others. For example, there are several useful

metrics for evaluating retrieval quality, while LLM evaluation is only partially solved and only for specific setups.

Loops and Branches

Simple LLM system designs, such as a basic RAG setup, are based on a unidirectional graph with a starting point (the input query) and an end point (the LLM's response). This linear graph has only one possible trajectory, as can be seen in the sketch in Figure 4-1: upon receiving the user query, it's used by the retriever to identify relevant documents from the database. Those documents are then embedded in the prompt along with the query and sent to the LLM, which generates an answer.

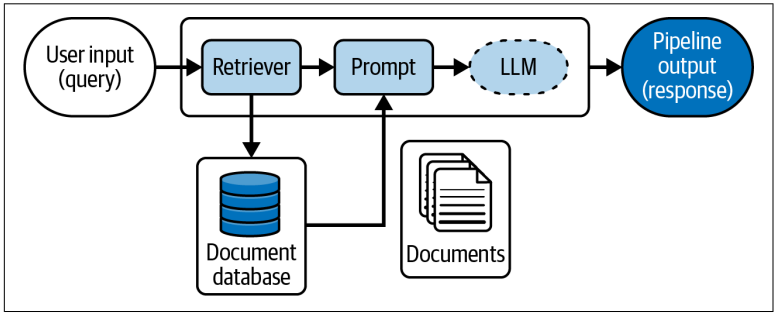

Figure 4-1. A sketch of a straightforward RAG pipeline

However, graphs can be much more complex, and LLMs have the potential to be used in more intricate setups. For example, graphs can contain branching structures where the same starting point can lead to several different paths. A common way to improve retrieval performance is to use a hybrid retrieval setup that combines two categorically different retrievers, whose results are combined before they're passed on to the LLM. This setup often includes one based on lexical similarity and another based on semantic similarity, resulting in a higher overall recall. Other pipelines use classifiers that examine the query or some other part of the pipeline and route it to different workflows, each designed to solve a specific task.

One particular AI system design that takes modularity to the next level is the agent. Agents are complex pipelines that can traverse multiple trajectories depending on the task. They make use of an LLM's reasoning abilities by employing it to develop

advanced response strategies for complex queries. Agents have multiple "tools" at their disposal, which they use iteratively to arrive at the optimal answer. They can perform autonomous planning and decision making, using the tools to accomplish a given task without explicit instructions. Agents use branching to navigate different paths and looping to potentially repeat actions. Another common component in agents is an implementation of memory—most notably, it is used by chatbots (also known as conversational agents) to keep track of a conversation. Figure 4-2 shows an agent setup with access to multiple tools. In the figure, at least one of the tools is connected to a database (thus, it can perform document retrieval on demand, if the agent's plan deems it necessary).

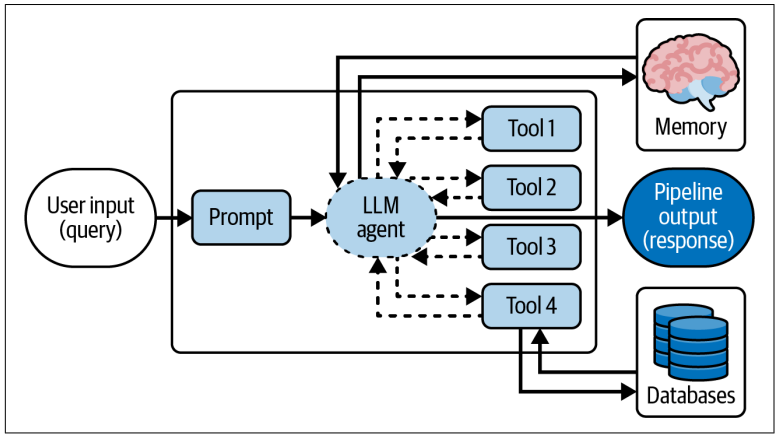

Figure 4-2. A sketch of an LLM agent that can use four tools and memory to solve a task

As part of their strategic planning capabilities, agents decide when they have completed the task and are ready to respond. Autonomous agents may be the closest we've come to artificial general intelligence (AGI). But for now, agents that go beyond the capabilities of regular chatbots are still too complex a product for most production use cases. And if it's hard to reliably evaluate LLMs, imagine how much harder it is for such a complex system with many moving parts that may be integrated with multiple LLMs. But at the rate we're seeing LLMs evolve, it's likely that the big agent moment is near.

Putting the Pieces Together

Now that we've learned about some of the key tools, skills, and knowledge that AI teams need to curate, it's time to regroup and see what we've learned about product development with AI. In the next chapter, I'll talk about why the approach I've outlined is essential for successful AI adoption in the enterprise, and what other aspects you need to consider to arrive at a mature, LLM-powered product.

CHAPTER 5

From Pilot to Product

An AI product that does not make it into production is not a product at all, but merely a demonstration of the technology's capabilities. There's no need to prove that LLMs can do amazing things—we've all experienced their skills firsthand. As I have emphasized throughout this report, the real challenge lies not in using these models to process your organization's data, but in integrating them seamlessly into a relevant business use case. Very few companies have been able to do this. With the obvious potential of LLMs, that will have to change.

AI teams that build their pilots with a business use case in mind are much more likely to see them evolve into mature products than teams that build narrowly focused IT demos. Product leaders must guide their teams through the thick and thin of the AI development cycle so that they can build robust, scalable solutions that evolve and thrive in a production environment. In this chapter, I offer a few final pieces of advice to help demystify this pilot-to-product journey.

Scalable Pilots

Most people who have never brought an actual AI product to production are unaware that the development cycle can realistically consist of hundreds of iterations. Each iteration can change the direction of the product, the user base it targets, the technology it uses, and the data it processes. A pilot therefore serves the dual purpose of proving the concept and laying the groundwork for future

product development. From there, you must iterate and experiment at scale to get something into production.

When designing your use case, it's important to avoid creating a pilot that is too narrow in functionality, as this will make any modification or scaling feel like an impossible task. Design the pilot with the ability to add new models, integrate parallel processing paths, or scale operations without requiring a complete system overhaul. A well-designed pilot should transition smoothly into a robust product, allowing for growth and evolution without sacrificing core functionality.

Getting Ready for Production

Once the prototype has been refined to the expected level of maturity, it is time to move it into production. This is where the solution is truly tested by time, users, and market demands. It's critical to have a solid framework for monitoring and managing the product after deployment; this is where MLOps comes in.

MLOps stands for machine learning operations. It extends the principles of DevOps to software projects that integrate machine learning. MLOps provides the tools to plan the software lifecycle holistically, from development to deployment to operations in production. The goal of MLOps is to create a sustainable and secure ecosystem for your AI-powered solutions—because deploying a complex ML-based product to production can be challenging. DevOps engineers are in high demand since they have the broad skill set needed to ensure the reliability, scalability, and performance of the system. This includes making decisions about the application's backend architecture, data storage, and computing power. However, there is one aspect that is unique to the deployment of LLM-based products that I'd like to focus on here: monitoring the performance of a generative language model.

The Challenge of Monitoring LLMs

To monitor an application in production, engineers define the metrics they want to track over time. These might include the number of requests processed in a minute, hour, or day, and the time elapsed between a request and a response. They might also want to monitor

the rate of negative feedback from users, or the topics that appear most frequently in their queries.

LLMs are much harder to monitor than simpler, deterministic machine learning models, such as sentiment classifiers, because they are so creative that it often requires a human to reliably judge their quality. Well-established NLP tasks like automated translation and summarization traditionally work with lexical metrics such as BLEU (Bilingual Evaluation Understudy) and ROUGE (Recall-Oriented Understudy for Gisting Evaluation) that quantify how similar the words of a machine-produced output are to that of a human-annotated one. The advent of LLMs, however, has intensified the search for robust metrics that can compare two textual sequences based on semantics rather than lexical similarity. In particular, RAG applications have proven to be particularly fertile ground for such metrics. Using a transformer model—the same neural network architecture that powers LLMs themselves, albeit on a much smaller scale—these new metrics measure the groundedness of a generated answer in the documents on which it is based. Combined with the well-established evaluation methods that have long existed for the retrieval step itself, such as recall and mean reciprocal rank (MRR), these semantic metrics allow us to get a fairly accurate idea of the usefulness and correctness of a given RAG application. In the future, I hope to see a lot more of these kinds of metrics for a variety of purposes in LLM setups.

Keep an Eye on AI Regulation

As the AI landscape continues to evolve, so does the regulatory environment surrounding it. With this new technology comes the responsibility to ensure that your operations comply with emerging legislation in relevant jurisdictions. A proactive approach to data governance involves security and legal experts who can assess potential risks and develop comprehensive mitigation strategies, especially when dealing with sensitive and proprietary information. It's also important to carefully review vendor contracts to ensure robust data protection. To protect their own data and that of their customers, companies should look for vendors that allow them to opt out of storing their prompt history and that can credibly demonstrate that the data submitted will not be used to train future models. Of course, excluding personally identifiable information (PII) from hosted LLM interactions is also critical. The path to

responsibly harnessing the potential of AI requires a comprehensive understanding of regulatory frameworks and a commitment to secure, ethical, and compliant operations. In addition to mitigating risk, this approach builds user confidence, which is crucial to the successful adoption of any new technology.

Building Meaningful AI Products

In this report, I hope to have shed some light on the unique challenges of AI development and helped you get a clearer idea of what it takes to build meaningful products with LLMs. Following the general release of the ChatGPT browser interface, organizations and individuals alike went through a phase that can be described as a mixture of fear, awe, disbelief, and dollar signs. As we enter a new phase of LLM adoption, it's imperative that thought leaders refocus their own approach to AI. Product people across all industries need to understand the tangible value this technology can bring to their organizations—and what they need to facilitate for it to become a reality.

You now understand the critical importance of developing a comprehensive use case—together with a cross-functional team of AI engineers, business representatives, and subject matter experts. You have seen that this new technology requires visionary and pragmatic product leads who know how to ask the right questions. You've seen that an agile MLOps workflow with frequent, small changes to the product is as important as ever in the age of generative AI. You've explored the importance of data best practices and monitoring LLMs in production using a mix of metrics. Finally, you've learned how to avoid the pitfalls of a nonscalable, narrowly focused IT demo that collapses at the slightest change or attempt to scale—and instead build scalable, visionary, and secure products that can meet the unique business needs of your organization.

About the Author

Isabelle Nguyen has a background in linguistics and a master's degree in NLP and machine learning. She currently works as a technical content writer at deepset. Her main focus is to demystify the intricacies of generative AI and LLMs and make these complex topics accessible to a wide audience.