数据科学与大数据技术

构建大模型数据科学应用：从机器学习升级到大模型

[美] 克里斯汀 • 科勒(Kristen Kehrer)
凯莱布 • 凯撒(Caleb Kaiser) 著

王奕逍 译

清華大學出版社
北 京

北京市版权局著作权合同登记号 图字：01-2014-5110
Kristen Kehrer, Caleb Kaiser
Machine Learning Upgrade：A Data Scientist's Guide to MLOps, LLMs, and ML Infrastructure
EISBN：978-1-394-24963-3

图书在版编目(CIP)数据

构建大模型数据科学应用 : 从机器学习升级到大模型 /
(美) 克里斯汀·科勒 (Kristen Kehrer), (美)凯莱布·凯撒 (Caleb Kaiser) 著 ;
王奕逍译. -- 北京 : 清华大学出版社, 2025. 3. -- (数据科学与大数据技术).
ISBN 978-7-302-68583-8

Ⅰ. TP391

中国国家版本馆 CIP 数据核字第 20251ST779 号

责任编辑：王　军　韩宏志
封面设计：高娟妮
版式设计：恒复文化
责任校对：马遥遥
责任印制：杨　艳

出版发行：清华大学出版社
网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：148mm×210mm　　**印　　张：**5.125　　**字　　数：**157 千字
版　　次：2025 年 5 月第 1 版　　**印　　次：**2025 年 5 月第 1 次印刷
定　　价：49.80 元

产品编号：107463-01

技术编辑简介

Harpreet Sahota 自称是一名生成式 AI 黑客，拥有统计学和数学专业本科与研究生学位。Harpreet 自 2013 年以来一直在数据领域工作，担任精算师和 ML 工程师，是生物统计学家、数据科学家，拥有统计学、机器学习、MLOps、LLMOps 和生成式 AI(重点是多模态检索增强生成)方面的专业知识。他喜欢钻研新技术，也与妻子 Romie、孩子 Jugaad 和 Jinda 共享着温馨的家庭生活。他的著作 *Practical Retrieval Augmented Generation* 将于 2025 年出版。

致　谢

写作本书是我们两人的一次愉快合作，我们有共同的愿景，得到一个令人难以置信的团队的支持，他们使所有想法变成现实。非常感谢 Wiley 团队，特别是 James Minatel 和 Gus Miklos，他们肯于奉献，专业知识过硬，将我们的手稿变成一本精美书籍。深切感谢技术编辑 Harpreet Sahota，他提供了宝贵的反馈意见，并帮助我们修改稿件，重新梳理思路，他的见解和指导对最终成书至关重要。衷心感谢各位读者，我们希望本书能为你的探索提供宝贵的见解，激发出新的想法。

前　言

欢迎你踏上现代 ML(机器学习)之旅，此次旅程将充满活力！过去，数据科学多应用于商业智能工作，而如今，处理数据的方式已经大相径庭，多使用前沿的多组件系统。

希望本书能让你爱不释手。本书没有罗列方法，也不是一本全面介绍 ML 的书籍。本书旨在讲述现代 ML 相关的挑战，将重点介绍数据版本控制、实验跟踪、生产后模型监控和部署，并提供代码和示例，以便你能立即上手。

第 1 章讲述基础知识，揭示管理机器学习的工作流程如何从 CRISP-DM 等传统的线性框架演变为 LLM(大语言模型)驱动的应用。强调需要利用一个统一的框架来构建基于 LLM 的应用。

第 2 章将带你见证一种端到端的 ML 方法，探索生命周期、生产级 ML 系统的原理和 LLM 应用的核心。

第 3 章阐述“以数据为中心”的观点，强调数据在现代 ML 中的作用。该章需要你动手练习，将创建 embedding(嵌入)并用向量数据库进行文本相似度搜索。将道德准则和数据版本控制策略结合起来，以确保你采取负责任的一体化方法。

第 4 章将引导你选择正确的 LLM、利用 LangChain 并微调 LLM 性能。

在第 5 章中，将组件组装在一起，从原型过渡到应用。该章还演示如何构建仪表盘和 API(应用程序编程接口)，使你的模型可为最终用户提供结果。

第 6 章将完成 ML 的生命周期，对模型进行监控、重训练管道，并规划未来的部署策略，分析如何与利益相关者沟通。

最后，在第 7 章中，回顾了在整个过程中总结的最佳实践，探讨了 LLM

的新趋势，并提供了资源供你进一步学习。

本书不仅是一本指南——它是一次冒险，是一次穿越现代 ML 风景区的邀约，也是一次为你配备导航工具，让你汲取知识的机会。所以，朋友们，系好鞋带，让我们踏上旅途吧！

下载示例代码

读者可扫描封底二维码，下载配套的示例代码。

目　录

第1章

现代机器学习简介

在过去 20 年里，数据科学在很大程度上聚焦于利用数据制定业务决策。典型的数据科学项目围绕数据收集、数据清洗和数据建模而展开，生成一个数据仪表盘，最后制作演示文稿与利益相关者分享成果。数据科学给企业带来大量收益。

传统上，我们将执行描述性分析以做出合理决策的项目称为商业智能(BI)。从理论上讲，商业智能是数据科学的一个特定领域。数据科学从技术上讲更宽泛，指将统计方法(包括建模)、编码和领域知识应用于数据的实践；而商业智能则更狭义，指采取数据驱动的方法做出业务决策，这些决策更多地关注描述性和诊断性分析，而非数据科学家执行的预测性分析。然而，我们认为所有分析师和 BI 专业人员都在数据科学领域工作。

在实践中，如果你近十年来担任过分析师或数据科学家，你必然以某种方式在商业智能上花费过大量时间。

许多人不认可上面这种说法，因为在传统意义上，商业智能属于“BI 分析师”等职位的范畴，而数据科学家的职责往往更加多样化，以研究为重点。这些人的想法有一定道理，但在一个普通的分析组织中，角色的职责和职能的界限并不那么明显，使得很难将数据科学与商业智能巧妙地分开，在处理数据时总有重叠。

假设你在一家普通公司担任分析师，你可能负责回答“发生了什么”的问题，通过描述性分析来获得过往业绩的快照；会用 Excel、SQL 和可

视化软件生成报告和仪表盘，会监控关键绩效指标(KPI)，并根据历史数据帮助做出战略决策。也可能在启动项目前预设置了过程中使用的 KPI、数据源和机器学习模型(如果有)，BI 分析师或业务分析师负责对这些进行管理。

当公司有全新的数据需要处理时，通常会将另一个仪表盘以自助服务的形式提供给非技术利益相关者(此时，可能发生应当使用 Tableau 还是 Power BI 的激烈争论)。通常而言，这是一种区别普通公司中数据科学家和 BI 角色的有效方法。数据科学家通常负责组织中更具技术性、研究更密集的分析项目：探索新的数据源、执行预测分析、完成假设检验、研究新的机器学习模型等。他们所做的大部分工作仍然属于商业智能的范畴。至少说，在过去是这样。

1.1 数据科学与商业智能渐行渐远

数据科学重在研究，在当今的机器学习时代，我们有必要强调数据科学的原理。在许多数据科学家从事的高级分析中，若要构建一个机器学习模型，那么该模型很可能只用于研究。特别在几年前，你不大可能在产品中部署或实现一个模型。你可根据客户行为对客户进行细分，或构建一个预测模型，以更深入地了解客户。利用有关客户的新信息，通过进一步的假设检验，可推动产品增加功能或更改功能。尽管如此，一旦研究成果传达给业务部门，模型本身就不会再使用了。

在过去，数据科学家并非不想训练影响庞大的模型，而是有心无力。2022 年，Gartner 发布了一项关于美国、德国和英国使用机器学习的公司的调查，这些公司的数据科学家在 ML 生态系统中历经多年开发了不少模型，但其中只有 54%投入生产。

数据科学家主要通过简单、实用的商业智能为企业贡献价值，而构建的模型则显得“华而不实”。但时至今日，这种情况开始发生变化。

现在，建模工作本身变得更加可行，在产品中的使用越来越普遍。涌现出 AutoML 等新工具，ML 框架也得到改进；数据科学家可以更方便地

在几乎任何类型的数据上训练有用的模型。迁移学习因计算机视觉而普及，大语言模型的爆炸式增长起到进一步的推动作用；在许多方面，更容易训练有影响力的模型。随着机器学习基础设施和生态系统的完善与发展，部署也变得更加简单。

因此，数据科学家越来越致力于为业务用例建模，与传统的 BI 工作渐行渐远。越来越多的数据科学家负责构建和监控机器学习模型。然而，这个转变过程给数据科学家带来一系列全新的挑战和责任；本书旨在帮助数据科学家走出以 BI 为中心的世界，走入以生产为中心的多组件机器学习系统的新世界。

本书介绍的许多原则可泛称为 MLOps(或 LLMOps)，需要明确的是，我们的目标不是成为 MLOps 工程师，因为应用管理工具和基础设施不是数据科学家的工作。相反，希望数据科学家通过理解 MLOps 原理，构建可靠、可扩展、可复用、有影响力的模型。

在深入探讨这些原则之前，我们应该花一些时间来了解机器学习的变化。

1.2　从 CRISP–DM 过渡到最新的多组件 ML 系统

自诞生以来，机器学习经历了漫长的发展过程。20 世纪 90 年代，出现了 CRISP-DM(CRoss-Industry Standard Process for Data Mining，跨行业数据挖掘标准流程)，用于描述数据建模项目的典型阶段。长久以来，CRISP-DM 一直是管理机器学习工作流的主导框架。CRISP-DM 使数据科学由零散变得井井有条。CRISP-DM 框架包括六个关键步骤。

(1) 理解业务需求：从业务角度定义项目的目标和要求。

(2) 理解数据：收集和探索数据，了解其质量和结构。

(3) 准备数据：清洗、转换和组织数据，使其适合机器学习。

(4) 建模：构建和评估机器学习模型以解决业务问题。

(5) 评估：评估模型在实现业务目标方面的性能。

(6) 部署：将模型集成到生产环境中。

在过去的数据项目中，模型只是管道的一个环节，用于生成特定的报告。CRISP-DM 框架比较简洁，步步推进，所以十分适用于此类项目。

然而，最新的机器学习系统更像流水线，而非管道。系统中有许多相互关联的组件，适于解决更广泛的问题，起码会带来以下技术挑战。

- 数据大小和种类：随着大数据的出现，机器学习项目通常涉及海量的不同类型的数据集，如文本、图像和结构化数据。需要采用新方法来处理这些不同的数据源。
- 复杂算法：随着深度学习、强化学习等的出现，机器学习算法越来越复杂，需要专门的工具和框架来实现与训练这些算法。
- 模型部署：现代机器学习系统需要持续更新和监控模型，使部署成为一个复杂的、持续的过程。
- 可扩展性：过去，每个月只需要生成一次报告。而现在，必须根据数千个并发用户的需要，实时地进行推断，这是一个巨大的挑战。
- 协作：机器学习团队通常由数据科学家、数据工程师和领域专家组成。必须配备协作工具和平台，来管理这些不同职能的人员。
- 机器学习伦理：机器学习对社会的影响越来越大，使伦理问题和治理实践成为更高的优先事项。确保公平性、透明度和合规性已成为机器学习系统的重要功能。

幸运的是，一些才华横溢的数据科学家和工程师多年来致力于研究和解决这些问题。在现代机器学习生态系统中，已经产生了大量用于应对这些挑战的工具。这里举一些例子：①数据版本控制、实验跟踪和跨团队协作解决方案；②模型注册表；③用于生产环境的模型监控工具；④数据湖和仓库(使我们能有效管理、存储和查询大量数据)；⑤机器学习框架(使建模变得更加容易)；⑥开源库(确保负责任地使用机器学习模型，确保合乎道德标准)。

在本书中，我们将探讨这些工具，并利用这些工具构建实际项目。在此之前，有必要花点时间讨论一下近十年来机器学习生态系统中最具变革性的转变之一：大语言模型。

1.3　LLM 提升了 ML 的能力和复杂度

大语言模型(Large Language Model，LLM)的出现极大地提升了机器学习系统的能力和复杂性。GPT-3、BERT 等模型及其后继者重新定义了自然语言理解和生成的能力上限。

这些模型规模庞大，预训练参数量大，能完成各种任务。例如，LLM 在自然语言理解(Natural Language Understanding，NLU)方面的表现具备了前所未有的性能，如情感分析、文本摘要、语言翻译和问答，样样精通。LLM 从根本上改变了自然语言处理(Natural Language Processing，NLP)的格局，可根据上下文，生成用于不同领域的不同风格的连贯文本；在 LLM 的基础上，内容生成工具、聊天机器人和 AI 辅助写作如雨后春笋般涌现。

> 预训练 AI 模型(通常是最先进的深度学习模型)一般在大型数据集上进行训练，以执行特定任务。可以使用任何类型的数据，包括图像、文本、音频、表格数据等，具体取决于用例。

此外，预训练的 LLM 已成为迁移学习的强大工具。通过微调特定领域的数据，就可将这些模型用于各种应用领域，减少了每个新用例需要的标记数据量。预训练的 LLM 不仅能处理文字，还能处理图像及音频。这为复杂的多模态应用(如图像字幕、视觉问答等)开辟了新途径。

当然，能力的提高必然伴随着复杂度的增加。例如，考虑以下常见的 LLM 任务。

- 理解语言：LLM 可理解语言的微妙之处，根据上下文随机应变，AI 系统(如聊天智能体)更聪明。然而，聊天智能体需要频繁地进行推理(即使用已训练的 AI 模型进行推理，揣摩对方的心思，进行预测，解决问题)。假设有 1000 个用户；针对每个用户，每隔几秒钟就要进行一次推理；如何使用包含 170 亿个参数的模型来并发地支持这些用户呢？

- 提取知识：LLM 可从非结构化的文本中提取出结构化知识，这在数据挖掘、信息检索和内容管理中得到广泛应用。但如何摄取和存储这些知识呢？如何使系统动态找出这些知识？
- 生成内容：LLM 可以根据上下文生成极富创意的内容，如诗歌、代码、一篇完整的文章，甚至艺术、音乐和文学作品。如何在尊重版权法的前提下生成内容？如何防止种族主义或其他有害输出？如何找出并删除错误信息？
- 多模态 AI：其他深度学习模型有卷积神经网络(Convolutional Neural Network，CNN)等；如果结合 LLM 与 CNN 来完成图像处理，将获得一个高级的、功能强大的多模态 AI 系统。这些系统能理解融合了文本、图像和其他类型数据的内容，也能生成这些内容。然而，必须协同地部署和管理所有这些模型，如何有效地做到这一点？

研发工作正在使 LLM 更高效、更可解释，而且减少了模型消耗的资源。随着 LLM 价格的下降，人工智能可能大众化，在各个领域催生更多创新成果。

在本书中，当探索机器学习的新世界时，将列举一个 LLM 项目进行演示。

1.4 你能从本书中学到哪些知识

本书是一本 MLOps 和 LLMOps 指南。如果你以前从事传统的 BI 工作，或有研究背景，那么本书就是为你准备的。后续各章将介绍各种工具和工作原理，并在此基础上构建项目。我们希望你在未来的项目中使用这些技术。总体而言，本书将涵盖以下内容。

- **数据管道和版本控制**。将介绍数据管道和版本控制的概念。这将确保数据得到一致处理，并可跟踪和管理对管道的更改。
- **实验和模型版本控制**。将扩展建模阶段，纳入以下事项：架构搜索和模型版本控制等。

- **持续评估**。将扩展评估阶段，纳入以下事项：持续监控模型性能，并实施自动检查，以检测性能随时间的下降状况。
- **CD(持续部署)**。将更新部署阶段，纳入以下事项：CD，快速部署更新的模型。但不添加 A/B 测试。
- **监控和模型重新训练**。将添加一个维护阶段来监控生产环境中的模型，根据需要在新数据上重新训练模型。
- **协作工具和工作流**。将推广协作工具和工作流，以促进数据科学家、数据工程师和运营团队之间的跨职能合作。

要在不断发展的数据科学和机器学习领域纵横驰骋，就必须理解 MLOps 原则。我们面临着各种挑战，从数据版本控制到模型评估和部署；为了应对这些挑战，就必须采用系统的、可扩展的方法。通过将数据管道、版本控制、实验、持续评估和协作工作流整合到你的项目中，可提高可重复性和可扩展性，并确保模型随着时间的推移保持有效性和适应性。正如前面提到的，我们将在一个基于 LLM 的项目的引导下讲述这些原则。应用程序将完成问答任务，将 YouTube 视频用作外部数据源。第 2 章将介绍项目的一些背景知识。第 3 章开始正式开发项目。

第2章

一种端到端的方法

本书的重点是构建一个端到端的可用于生产环境的机器学习系统。我们应该首先确定一些术语的含义。在过去20年里，“端到端”和“生产”等术语在数据科学领域被广泛使用，而且这些术语的含义也与时俱进。

假设时间回到2015年，你在一家鞋类零售商的BI团队工作，当时处理数据的方式与今天大相径庭。你的团队负责预测下一季度的销售额。端到端系统会是什么样子？

你首先构建数据集。在2015年的时候，要想访问和整理公司的数据堪称一场噩梦，你的团队只有付出相当大的努力，才能整理出一个干净的数据集。接下来，你将专注于建模。团队可能尝试使用各种模型，从ARIMA到随机森林，甚至梯度提升(毕竟，XGBoost嵌入在2014年就发布了)。经过大量的调整和优化，并经过一些可靠的验证，你终于得到了模型。现在，可开始预测下一季度的销售额并与其他人分享你的结果了。你可能已经为CRO(首席收入官)制作了一个仪表盘，或者每天使用SPSS(Statistical Package for Social Sciences，社会科学统计软件包)等工具手动进行预测。也许你设置了一个每天都会运行的作业，通过宏来创建新一天的实际值。或者，你每天上班的第一件事就是检查模型的预测值，写一封电子邮件来分享结果。

从很多方面讲，预测销售额项目比较直接，但并不容易。由于需要执行如此多的手动操作，这样的项目都是困难的。很难从多年遗留的杂乱数

据中整理数据集。向非技术受众展示你的预测结果而不让他们感到无聊是一门艺术。在建模阶段，你可能完成大量实验，执行可靠的验证也很费事。但是，若将这个项目分成多个组件，就不需要做出那么多架构决策了。

- 数据摄取：公司的数据存储方式起着决定性作用，但你需要决定如何摄取数据并生成数据集。
- ML 框架：你可能使用 scikit-learn(一个用于建模的 Python 库)来构建模型，但因为当时是 2015 年，你也可以使用统计工具或团队构建的一些内部框架。
- 可视化库：如果公司使用了特定的仪表盘解决方案，你会使用它。否则，你将使用任何自己喜欢的库或 Excel 来生成报表和图表。

大致如此。你不需要做更多的架构决策。当时是 2015 年，除了电子表格，不存在任何真正的实验管理解决方案。数据版本控制不太可能以正式方式完成，模型也不必部署。你可能通过运行 notebook 或本地脚本(宏)来生成预测，这些脚本可能与某种版本控制一起存储。这基本上就是端到端系统包含的全部内容，而且是有效的——至少可用于特定的系统。

但是，如果是一个更复杂的机器学习系统，如 YouTube 搜索助手，又会怎样呢？该系统需要多个模型在管道中交互，涉及一个存储着嵌入(embedding)的向量数据库。

> 向量数据库专为存储和查询高维数据而构建。许多流行的技术，如 RAG(Retrieval Augmented Generation，检索增强生成)，都依赖于操纵文本嵌入；嵌入是存储在向量数据库中的高维向量。

ML 应用程序必须实现一些检索逻辑来获取相关的视频和摘要，这是一种从视频中生成文本记录(transcript)并为文本创建嵌入的方法。需要能够实时访问推理管道；ML 应用程序的功能必须是完备的，前端不能仅是某个报告中的图表。当然，你的系统需要能够扩展以支持大量的并发用户。

最重要的是，更复杂的机器学习系统并非只执行一次数据分析，而是一个持续的软件项目，需要维护、监控和持续改进。

在本章中，我们为设计这样的机器学习系统提供一个框架。首先详细讨论一下 YouTube 搜索助手。

2.1　YouTube 搜索智能体的组件

首先，大致描述一下我们的系统。当用户输入问题时，系统会在 YouTube 上搜索相关视频，然后将视频的文本记录添加数据库中；数据库也越来越大。此后，系统根据为用户搜索查询创建的文本记录，从整个数据库中提取出最相关的文本记录，并传递给语言模型。在实际中，最终结果如图 2.1 所示。

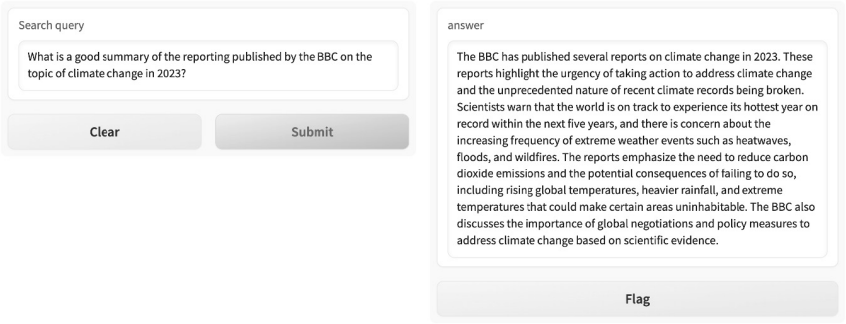

图 2.1　YouTube 搜索查询结果

这里的示例模型未对 2023 年的数据进行训练，而使用 RAG 与语言模型共享新信息。本书后续章节将详细讨论 RAG。

接下来仔细分析该项目的不同组成部分。组件大致可分为以下几类。

- YouTube 搜索：有一个运行 YouTube 搜索并获取相关视频的系统。然后，我们从这些视频中生成文本记录(transcript)。
- 存储嵌入：使用嵌入模型将文本记录块转换为嵌入，将嵌入存储在向量数据库中。再使用相同的嵌入模型将用户的初始问题转换为嵌入，执行相似度搜索，以从视频中检索最相关的摘录。最后，返回与嵌入和输入上下文相关的文本，完成最终的 LLM 推理。

- 大语言模型：在整个系统的关键点上使用 LLM。使用一个模型将用户的问题转换为相关的 YouTube 搜索，完成最终的问答任务，并生成嵌入。
- 用户界面：我们在 ML 应用程序中接收用户输入并显示输出。

每个类别中又有许多需要设计和实现的单独组件。图 2.2 展示了主要组件。

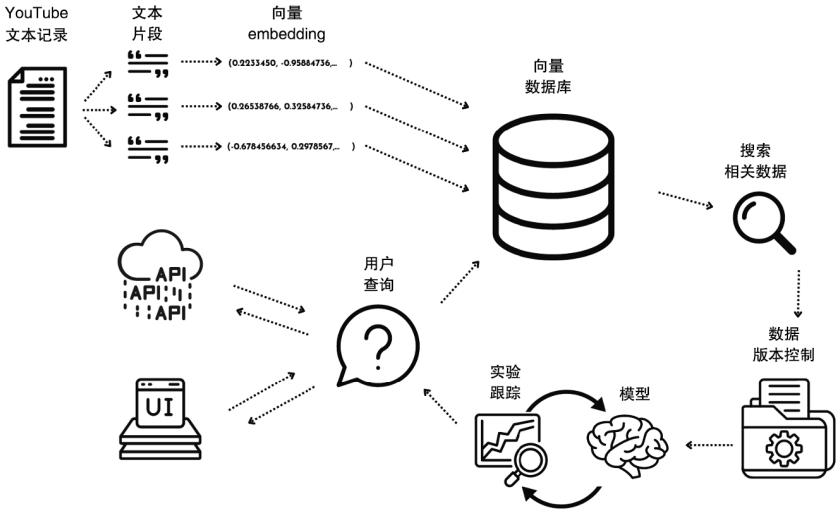

图 2.2 YouTube 搜索的组件

有必要了解这个系统中不同组件的依赖关系。你嵌入的文本决定着嵌入的质量，这意味着文本记录系统必不可少。同时，如果找不到相关的视频，文本记录再完美也毫无意义，这意味必须有一个卓越的 YouTube 检索系统。

将根据项目的需要做出许多设计决策。例如，如果你需要微调模型，那么可供使用的潜在 LLM 架构将大大减少，因为许多最流行的托管 API 不允许微调，而且许多主流模型架构本身的微调成本极高。同样，在这样的系统中，你也不必过度考虑关于基础设施的许多决策。

在这种支持大量并发用户的系统中，需要考虑哪些事项呢？如果系统输出的质量变差，你如何获悉呢？一旦了解到这一点，你会从哪里开始调

试？如果配备了一个数据科学家和工程师团队来改进这个应用程序，你如何将应用程序输出的变化与系统中的某个特定变化联系在一起呢？

2.2　生产中使用的 ML 系统的核心原则

架构是软件工程中的一个棘手主题，主要是因为没人真正确定它是什么。宽泛地讲，当人们讨论一个系统的基本逻辑而非实际的实现代码时，倾向于提到“架构”。这些讨论往往过于冗长，生成了大量图表，提出不少分类法和方法论，而实际构建者却可能忽视架构的要求。

然而，不能因此就说架构可有可无，我们只能从更贴近实战的角度去理解它。我们非常喜欢 Ralph Johnson 和 Martin Fowler 给出的定义：“架构就是最重要东西，不论它具体是什么。”

本着这种精神，我们希望专注于设计机器学习系统时我们认为“重要的东西”。我们不准备进行长篇大论，也不准备提出一个僵化的方法论，而是分享一套设计机器学习基础设施的核心原则，并给出遵循这些原则生成的示例代码。

对于生产中使用的 ML 系统，需要强调和注意以下几点。

- **可观察性(Observability)**。当系统输出的质量开始下降时，必须能够注意到这一点，并进行反向推导，找到质量下降的起因。不仅需要进行监控，还需要记录和显示数据。可观察性涉及对系统内部的实际分析。在 ML 环境中，这要求模型具有一定程度的可解释性。例如，当问答任务因为生成上下文的上游问题(而非因为问题过于复杂，模型无力处理)而失败时，会看到哪些迹象？可观察性对于分析根本原因来说是无价的。
- **可再现性(Reproducibility)**。机器学习系统本质上是概率性的，很难明确地说明特定变化是如何影响整个系统的；这也意味着，在同一数据上训练模型，可能得到不同的结果。为对我们做出的任何决定充满信心，需要能够重现结果，这意味着，需要对构成系统的所有不同参数和内部状态进行强有力的跟踪与版本控制。

- **互操作性(Interoperability)**。有人并不认可这是一个通用原则。毕竟，有很多公司运行的软件栈虽然在其生态系统以外无法互操作，但仍然健壮、稳定。然而，在 ML 领域变化如此之快的情况下，互操作性至关重要。你可能希望顺应 ML 领域的一些重大变化，而拥有一个可互操作的平台使这变得更加容易。
- **可扩展性(Scalability)**。规模必然是生产中机器学习的问题之一。即使对于较简单的机器学习系统，支持数千个并发用户也需要占用大量的计算资源。如果系统在设计时未考虑这一点，成本很容易失控(或者 ML 应用程序可能走向失败)。
- **可改进性(Improvability)**。ML 系统必须以一种可改进的方式进行构建。这听起来十分简单，但说易行难。以本章开头的销售额预测为例，在完成所有模型拟合、参数优化和实验以实现模型性能的最大化后，你如何才能改进模型？另一位数据科学家能否在六周后启动该项目，并在不推倒重来的前提下进行有意义的改进？能否找到训练代码？最初的创建者可能尝试了各种变量组合和不同技术，而新的建模者很可能对此不知情，无法加以改进。

若能将以上各项的英文单词的首字母合成一个新单词，那该有多好。ORISI 是斐济人的姓氏、一个意大利葡萄品种、印度奥里萨邦的一个词语，都不是非常知名。如果其中任何一个让你觉得与软件基础设施特别相关，尽管使用。接下来简单分析一下，这些原则是如何应用于实际的 YouTube 系统的。在后续章节中，将列举实际的编码示例，更深入地探讨每个组件。

2.2.1 可观察性

可观察性用于衡量我们从外部输出评估系统内部状态时能有多高的准确性。换句话说，如果查看系统的总体输出，包括它生成的答案，以及日志、警报或其他分析，我们能很好地理解系统内部发生的事情吗？

可无限地对可观察性进行优化。在本书的各个项目中，在讲解技术时，将尽量追求简明易懂，同时追求生产环境中问题的可观察性，在两者之间取得平衡。

对于 YouTube 搜索助手(也称 YouTube 搜索智能体)而言，我们在可观察性方面的第一个突破是使用 OpenLLMetry 这样的工具来自动跟踪与模型的交互链。模型在流程每个步骤的行为都留下可追溯的快照，包括提示、响应和元数据。这样，系统可以发出一些简单的警报。

如果系统的输出质量开始下降，我们就可以利用这些可观察性特征，真正地查看工作流每个阶段的提示和响应。通常能根据这些数据立即诊断出性能恶化的原因。

2.2.2　可再现性

让我们想象一下，你已经诊断出，构建上下文窗口的方式存在潜在的问题。你进行了修复，事情似乎进展顺利。如何确定真正解决了这个问题？也许以前有几次搜索有些怪异之处，最近的搜索却是正常的，你的“修复”实际上并未改变任何东西。

为确认问题，你需要重现以前会返回糟糕结果的搜索。前几次搜索时，用到一些数据、参数和提示，为此，你需要访问完全相同的信息。参数和提示由可观察性平台保留，但我们需要额外的重现工具。

可用于对机器学习数据进行版本控制的工具有多种，你可从中选择一种；这样，在测试更改时，系统可使用相同的数据。下一章将深入探讨此类工具。

2.2.3　互操作性

有时，需要更改系统中的组件，使系统更趋完美。也许一家新公司发布的基础模型比 OpenAI 更好，也许你想构建自己的 LLM，也许你开发了一个更好的视频搜索系统。无论如何，机器学习的新技术不断涌现，如果你想跟上时代的步伐，拥有一个能适应这些变化的系统是十分重要的。

有许多书籍介绍如何构建具有互操作性的软件；本书主要讲述机器学习，不会深入探讨软件架构原理，仅重点介绍一些可行的原则。第 5 章将进一步充实这些内容。但此处要提醒你，关键是模块化。

实现系统时，最好将每个基本操作都独立出来，包含在各自的功能块

中。原则上，每个功能和每个工具只负责做一件事。这允许我们在不发生级联故障的情况下交换各个组件。例如，在后续章节中，将编写一个get_completion()方法来调用 LLM。若想更改基础模型，只需要编辑该函数，其他一切都不需要变动。同样，可观察性平台可将数据输出到任何主机，这意味着可随意更改分析平台，而不会中断系统。将模型投入生产时，数据互操作性同样构成一个巨大挑战。手机和手机应用程序通常具有高度的互操作性，不同设备和平台可以无缝通信、共享数据和协同工作，但在工业领域，尤其是医疗保健领域，情况往往并非如此。另一个部门可能使用不同的系统，甚至可能是落后的遗留系统，数据的格式和结构也可能是不同的；因此，各个部门的数据源互通成为问题。人们通常称之为“数据孤岛”，这可能是将机器学习模型投入生产的巨大障碍。

2.2.4 可扩展性

关于可扩展性，我们想回答两个主要问题：系统能否支持大量并发用户？能否在预算不超标的情况下支持大量并发用户？

为说明第二个问题的重要性，接下来来分析一个真实例子。2019 年底，一位名叫 Nick Walton 的机器学习研究人员分享了他通过微调 GPT-2 构建的一款基于人工智能的地牢爬虫游戏。该游戏可通过 Jupyter notebook 运行，Jupyter notebook 通过 Google Colab 共享。当时，Colab 为所有用户提供免费 GPU，因此 Nick 认为这是一种免费的游戏扩展方式。

这款游戏令人着迷，迅速走红，吸引了成千上万的玩家。玩家都在运行 Colab notebook，Colab notebook 从 Nick 的存储中下载模型(大小为 5GB)。数据传输不是免费的，托管这款“免费”游戏的成本攀升到每天 10 000 美元以上。实际上，对于 Nick 和他的合作者来说，将游戏作为一个完整应用程序发布，并将模型部署到 AWS 上的成本更低。

云计算为大众提供了运行大型模型所需的计算资源；有些公司运行的系统的推理和训练成本十分高昂，如果自建基础设施，可能造成公司破产，而若借助云计算，这些公司的系统将能正常运行。对于本书接下来几章介绍的 YouTube 搜索智能体项目，我们将注意确保系统能够以可管理的方式

扩展。例如，向量数据库在 Kubernetes 集群上运行，从而能够快速扩展。为简单起见，将使用维护者 Zilliz 慷慨提供的免费托管计划。该数据库在一定范围内是开源的；如果系统规模太大，超过了免费上限，我们可快速计算出自行托管数据库是否更合算。虽然我们知道，由于我们发送的流量不大，OpenAI 服务器不太可能崩溃，但我们知道，OpenAI 的 API 不是免费的，而且 OpenAI 那边也可能出现变故，例如可能无征兆地罢免其首席执行官；因此，我们已将系统与特定的 LLM 解耦。如果成本超出我们的预算，可探索用更便宜的模型，甚至托管我们自己的模型，而不需要重构整个系统。

2.2.5　可改进性

与机器学习系统相关的可改进性主要体现在两个方面。第一个方面与常规的软件工程思想是相通的：代码是否便于重构以改进系统？这又回到了关于互操作性和可扩展性的观点。我们的系统可以接受任何零散的改进，而不会造成大规模中断。

可改进性的第二个方面更适用于机器学习领域(尽管它在其他一些软件项目中也很常见，特别是那些涉及网络的项目)：系统在自我改进方面做得好吗？

你可能觉得，第二个方面让你摸不着头脑。这里以我们的搜索管道为例予以解释。对于给定的查询，系统将只查询有限数量的新视频。然而，一旦这些视频被查询出来，就会永久输入向量数据库中；相似度搜索是针对整个数据库进行的，不仅仅是我们最近查询的视频。因此，使用我们系统的人越多，数据库中存储的相关视频也越多；当用户使用同一数据库时，系统在未来查询中的表现将更好。在我们未干预的情况下，该系统正在自我改进。

这一原则尤其适用于涉及重新训练管道的系统。在此类系统中，提供模型性能反馈的用户越多，重新训练管道的影响就越大。

2.2.6 关于工具的注意事项

在本书中，我们将使用许多不同的第三方工具和库。这是无奈之举，如果从头开始实现这些工具，本书会变成丛书。当然，这确实为一些潜在问题找到了解决的办法。例如，如果你读本书时，一个工具不存在了怎么办？考虑到机器学习的发展速度，这并非天方夜谭。

为避免这种情况，本书一直谨慎选择符合可观察性、可再现性、互操作性、可扩展性和可改进性原则的工具。本着这种精神，下面列出选择工具时的一些标准。

- 不选择端到端平台。没有一个工具可以决定在本书中构建的项目的成败。
- 优先选择开源工具。使用的大多数工具都是开源的，在你阅读本书时，这些工具应当都是可用的。即使在使用云供应商的情况下，也试图与核心产品开源的供应商(如 Zilliz)合作。将使用 Comet 进行数据版本控制和实验跟踪。尽管 Comet 的核心产品不是开源的，但有一个强大的社区版本，维护着许多开源项目。
- 尽量使用简单的 UI。避免使用具有专用 API 或特定工作流的工具。例如，要求可观察性平台只使用与大多数其他 Python 日志框架类似的 Python 语法。

本章介绍了总体思路和核心原则，第 3 章将开始挖掘一些实际项目。