第2章 LlamaIndex 生态概览

相信读者已经对大语言模型有了深入的理解，并了解了它们的能力以及局限性。在本章中，我们将探讨如何利用 LlamaIndex 实现检索增强型生成 RAG，以弥合大语言模型在通用知识和个人私有数据之间的差距。

本章将涵盖以下主要内容。

- 优化语言模型——微调、RAG 和 LlamaIndex 的关系。
- 渐进式揭示复杂性的优势。
- 实践项目——个性化智能辅导系统 PITS 简介。
- 配置开发环境。
- 熟悉 LlamaIndex 代码仓库的组织结构。

2.1 技术需求

要完成本章的内容，读者需要准备以下工具。

- Python 3.11：`https://www.python.org/`。

- Git：https://git-scm.com/。
- LlamaIndex：https://github.com/run-llama/llama_index。
- OpenAI 账户和 API 密钥[①]。
- Streamlit：https://github.com/streamlit/streamlit。
- PyPDF：https://pypi.org/project/pypdf/。
- DOC2Txt：https://github.com/ankushshah89/python-docx2txt/blob/master/docx2txt/docx2txt.py。

本章的所有示例代码都可以在本书配套的 GitHub 仓库的 ch2 子文件夹中找到：https://github.com/PacktPublishing/Building-Data-Driven-Applications-with-LlamaIndex。

2.2　优化语言模型——微调、RAG 和 LlamaIndex 的关系

在第 1 章中，我们介绍了基础大语言模型存在的一些固有限制，例如其知识更新滞后，且可能生成不合理或不符合常识的回答。我们也初步探讨了 RAG 作为一种可行方案用于改善这些问题。通过结合提示工程技术和程序化手段，RAG 能够在一定程度上弥补大语言模型的不足。

什么是提示工程

提示工程指的是构造能够被生成式 AI 模型有效理解和处理的输入内容。这些提示通常以自然语言形式呈现，用于明确告知 AI 需要完成的具体任务。我们将在第 10 章对此进行详细探讨。

① 译者注，对于国内的读者，建议使用 DeepSeek 提供的 API，申请地址为 https://platform.deepseek.com/ api_keys。

2.2.1 RAG 是唯一的解决方案吗

当然不是。另一种方法是对 AI 模型进行微调，即利用专有数据对大语言模型进行额外训练，并嵌入新数据。这种方法是在一个通用数据集预训练过的基础模型上，继续进行更专业的数据集训练。这个专业数据集可以根据感兴趣的特定领域、语言或任务进行定制，最终得到一个既能保持广泛知识又能掌握特定领域专业知识的模型。图 2.1 展示了微调过程的示意图，供读者参考。

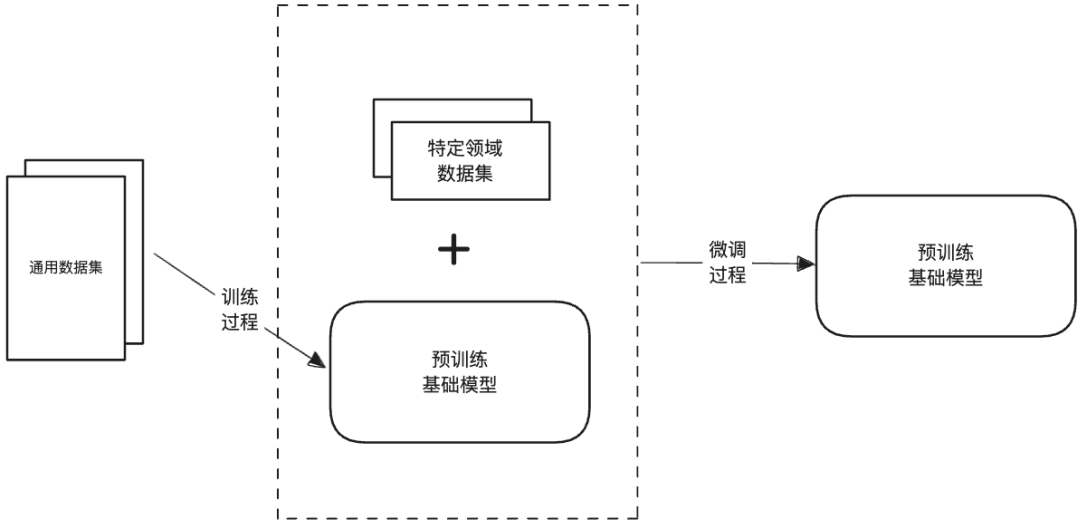

图 2.1 大语言模型微调过程

微调可以提高性能，但也存在如成本高、需要大量数据集，以及难以更新信息等缺点。此外，微调还会永久性地改变原始 AI 模型，这使得它不适用于个性化场景。这里，可以将原始 AI 模型比作一道受欢迎菜肴的经典配方，微调就像是根据特殊口味或需求修改这个配方。虽然这种改动可以让菜品更适合部分人群，但同时也改变了原有的配方。

注意

并非所有的微调方法都会永久改变基础 AI 模型：例如，低秩适应（low-rank adaptation，LoRA）是一种适用于大语言模型的微调技术，相较于传统的全微调方法，它更为高效。在全微调中，神经网络的所有层都会被优化，这种方式虽然效果显著，但非常耗费资源。而 LoRA 仅对两个较小的矩阵进行微调，这两个矩阵近似于预训练大语言模型的较大权重矩阵。采用 LoRA 方法时，原始模型的权重被冻结，这意味着在微调过程中这些权重不会被直接更新。模型行为的改变是通过引入这些低秩矩阵实现的。这样既保留了原始模型，同时又能使模型适应新的任务或提高性能。有关该方法的更多信息，请参见如下链接：`https://ar5iv.labs.arxiv.org/html/2106.09685`。

尽管 LoRA 在内存使用上比完全微调更有效率，但其实施和优化仍需计算资源和专业技能，这对某些用户来说可能是障碍。若要为众多用户提供个性化体验，则需要为每位用户重新执行调优过程，这样做并不具有成本效益。

这并不是说 RAG 比大语言模型微调更优秀。事实上，RAG 和微调是相辅相成的技术，常常联合使用。不过，为了快速接入变动的数据和实现个性化需求，RAG 显得更加合适。

2.2.2　LlamaIndex：构建可注入数据的大语言模型应用

LlamaIndex 允许读者快速构建出适应特定场景的大语言模型应用。通过注入目标数据，读者可以利用大语言模型提供准确、相关的结果，而不是仅依赖其预训练的知识。LlamaIndex 提供了一种简便的方法，可以将外部数据集与诸如 GPT-4、Claude 和 Llama 等大语言模型连接起来，从而在个人专有知识与大语言模型的强大功能之间架起一座桥梁。

注意

LlamaIndex 框架由普林斯顿大学毕业生兼企业家 Jerry Liu 于 2022 年创立，在开发者社区中获得了广泛关注和认可。LlamaIndex 不仅使得用户能够充分利用大语言模型的强大计算和语言理解能力，而且确保其输出基于特定且可靠的数据。这一独特优势使得

企业和个人都能从 AI 投资中获得最大收益，即通过同一核心技术实现多样化的专业场景应用。

例如，读者可以创建一个公司文档集合的索引。当读者提出与业务相关的问题时，结合 LlamaIndex 增强的大语言模型将依据真实数据提供答案，而非含糊不清的回复。

这样，你既能利用大语言模型的强大表达能力，又能显著减少错误或无关信息的产生。LlamaIndex 引导大语言模型从提供的可信数据源中获取信息，这些数据源可以包含结构化和非结构化数据。实际上，LlamaIndex 这个框架能够处理几乎所有类型的数据源。

如果读者还没有考虑过这个框架的各种潜在应用，这里简单列举几个想法，借助 LlamaIndex 可以实现如下功能。

- **构建专属的文档搜索引擎**：这一功能允许索引所有类型的文档，包括 PDF、Word、Notion 文档、GitHub 仓库等。索引建立后，你可以查询大语言模型搜索特定信息，从而打造一个专门为你的资源量身定制的高效搜索引擎。
- **创建具有定制知识的企业聊天机器人**：如果你的企业有特定的术语、政策或专业知识，可以让大语言模型理解这些细节。这样一来，聊天机器人既能处理基础的客户服务问题，也能应对通常需要人类专家处理的专业化查询。
- **生成大型报告或论文的精要摘要**：如果经常处理大量文档或报告，LlamaIndex 可以让大语言模型学习文档内容，并根据要求生成简明扼要的摘要，以突出关键点。
- **开发支持复杂工作流的智能助手**：通过在大语言模型上训练与组织相关的多步骤任务或程序，可以将大语言模型转化为智能助手，提供有价值的洞察和指导。

此外，图 2.2 展示了使用智能 RAG 策略降低特定领域模型微调成本的效果。

在深入探讨 LlamaIndex 框架的应用场景和实例之前，我们先来介绍该框架的架构及其设计原则。

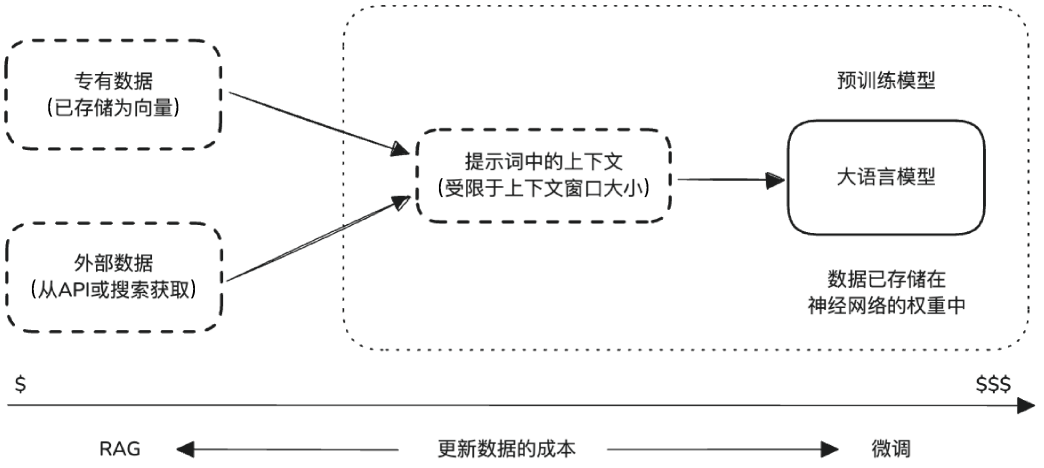

图 2.2　预训练大语言模型进行数据更新的相对成本

2.3　渐进式揭示复杂性的优势

LlamaIndex 的创建者希望这一工具能够被所有人轻松使用——无论是初探大语言模型的新手，还是打造复杂系统的专家级开发者。为此，LlamaIndex 遵循了“渐进式揭示复杂性”的设计原则。这个名字听起来可能有些花哨，但它本质上是一种设计理念：从简单起步，根据需要逐步引导用户接触更复杂的功能。

初次体验 LlamaIndex 时，读者将感受到如魔法般的便捷体验。仅需数行代码，读者就能完成数据连接并启动对大语言模型的查询。在内部实现上，LlamaIndex 负责将数据转换成高效索引，以便大语言模型能够利用它们。

查看下面的例子，它非常简单，只有 6 行代码。这个例子演示了如何使用 LlamaIndex 从本地目录加载一组文本文档，随后基于这些文档构建索引，并通过自然语言查询获取文档摘要视图。

```
from llama_index.core import VectorStoreIndex, SimpleDirectoryReader
documents = SimpleDirectoryReader('files').load_data()
index = VectorStoreIndex.from_documents(documents)
query_engine = index.as_query_engine()
response = query_engine.query(
    "summarize each document in a few sentences"
)
print(response)
```

注意

该示例仅用于演示目的。在实际运行前，我们仍需完成相应的环境配置。稍后将对此进行详细说明。

随着对 LlamaIndex 的使用增多，读者会发现更多强大的功能。LlamaIndex 提供了大量可调参数，允许选择针对不同用途优化的专门索引结构，对不同的提示策略进行详细的性能评估，自定义查询算法等。不过，LlamaIndex 始终以温和的方式开始，逐步引导用户深入了解其工作原理，对于快速和简单的项目，并不需要过多地深入细节。这种方式确保无论是初学者还是专家都能充分利用它的多功能性和强大能力。

接下来将简要介绍本书中的 LlamaIndex 实践项目，并为后续编写代码做好准备。

在深入学习本书内容并尝试练习其中的示例时，读者需要特别留意一个关键问题：成本控制。默认情况下，LlamaIndex 框架配置为调用 OpenAI 提供的 AI 模型。尽管这些模型具备强大的语言理解和生成能力，但其调用会产生一定的 API 使用成本。本书展示的许多 LlamaIndex 功能，如元数据提取、索引构建、检索及响应合成，均依赖于这些模型或嵌入模型。为了尽量减少这些不必要的开销，书中示例尽可能采用了简单的小规模数据集。

注意

强烈建议读者密切关注 OpenAI API 的使用情况，并通过以下链接查看和监控：`https://platform.openai.com/usage`。同时建议读者在隐私方面保持谨慎。API 使用和隐私问题将在第 4 章和第 5 章中进行更详细的讨论。

此外，如果读者想避免外部大语言模型带来的成本以及潜在隐私风险，可以参考第 9 章中描述的方法。不过需要注意，书中所有示例均是基于 OpenAI 提供的默认模型编写并测试的。因此，在本地部署环境中，部分功能可能表现不一致，甚至无法如期运行。

2.4　实践项目——个性化智能辅导系统 PITS 简介

动手实践是掌握技术最有效的学习方式之一。

为了帮助读者深入理解 LlamaIndex 的实际应用，本书设计了一个完备的实践项目——个性化智能辅导系统 PITS，一个能够互动式辅导用户学习新概念的 AI 导师。下面介绍 PITS 的工作原理。

第一步：自我介绍。初次使用 PITS 时，用户可以描述其希望学习的主题，并设定个性化的学习偏好。

第二步：上传资料。接下来，用户可上传与学习该主题相关的已有资料，例如 PDF、Word 文档或文本文件等，PITS 将接收并处理这些资料。

第三步：初步评估。系统将基于用户上传的学习资料自动生成测验。用户可选择完成该测验，以帮助 PITS 评估其对目标主题的理解程度，并据此动态调整学习路径。

第四步：生成学习资料。随后，PITS 将为用户量身定制学习资料，包括每张幻灯片的配套讲义和讲解文本，并将内容结构化分章，形成系统化的培训课程。

第五步：启动学习流程。个性化的学习过程由此正式开始。系统将根据用户设定的偏好逐步推进课程进度，确保每次学习内容均与其当前知识水平相适应。

第六步：互动学习。每当一个概念讲解结束，用户可提出疑问或请求补充示例以加深理解。PITS 将实时响应问题，生成新的测验内容，进一步解释关键概念，并根据用户反馈动态调整教学策略。

第七步：持续记录与回顾。用户与 PITS 辅导智能体的所有对话内容将被完整记录。它不仅能够记住用户的历史提问，还会保留自身的回应，避免信息重复，并确保学习过程的

上下文连贯性。

在学习的过程中，如果用户感到疲惫，可随时选择暂停。重新开始时，PITS 将会自动从上次中断的位置恢复，并简要回顾此前的学习内容要点。

图 2.3 展示了 PITS 系统的整个工作流程，以便更直观地理解其运行机制。

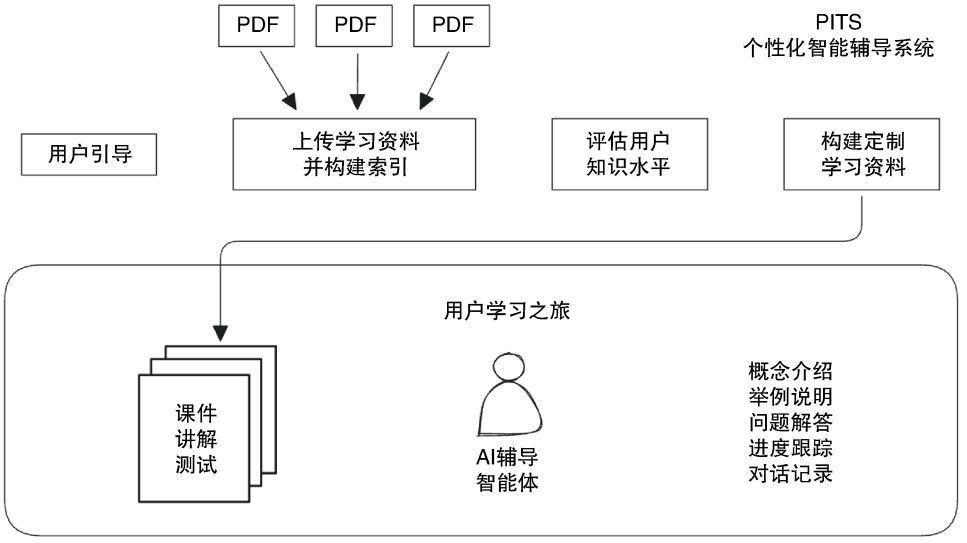

图 2.3　PITS 工作流程概览

这一流程高度体现了个性化原则，构建出近似“专属导师”的终极学习体验。

显而易见，要实现上述流程，PITS 需要在多个方面展现出高度智能性，主要包括：

- 能够解析并索引用户上传的学习资料。
- 能够与用户流畅对话，并持续记录对话上下文。
- 能够利用已索引的知识，实施有针对性的个性化教学。

在实现的过程中，LlamaIndex 发挥了关键作用，导入并索引用户提供的多种培训资料，包括培训手册、幻灯片，甚至是学生笔记和练习题等。而 GPT-4 则承担了与用户的交互式教学任务。可以说，LlamaIndex 所提供的知识增强能力是整个系统智能行为的基础。

PITS 让我们拥有一个量身定制的 AI 导师。

作者注

不确定你是否读过我的传记，我是一名培训师。当我第一次接触生成式 AI 并了解到 GPT-3 的强大之处时，我立刻意识到未来必将涌现出类似 PITS 这样能够辅助个性化学习的智能系统。随着 RAG 技术和 LlamaIndex 等工具的发展，这一构想正在迅速变为现实。我坚信，这类系统将为全球各地的人们提供更普惠的高质量教育资源，无论他们的位置、背景或经济状况如何。

接下来将开始准备 PITS 项目的开发环境，为后续的实践操作做好准备。

2.5　配置开发环境

在开始 LlamaIndex 的编码实践之前，正确配置开发环境是必不可少的前提。这一准备工作将确保读者能够顺利运行书中的示例代码和练习内容。

注意

为了保持简单性和一致性，本书的示例代码默认在本地 Python 环境中运行。虽然许多开发者习惯使用 Google Colab 或 Jupyter Notebooks 等在线平台，但部分示例可能不兼容。敬请读者谅解。本书的目标是最大限度地简化环境设置，让大家专注于核心内容学习，而非环境配置细节。

接下来，让我们快速配置计算机，准备开启激动人心的 LlamaIndex 编码体验。

2.5.1　安装 Python

本书建议读者使用 Python 3.7 或更高版本。推荐安装 Python 3.11 以获得更佳性能。

若尚未安装 Python，请访问官网 https://www.python.org 下载并安装。如果已安装旧版本，读者可以选择升级或并行安装更新的版本。

关于编程环境，作者个人偏好的编辑器是 NotePad++（https://notepad-plusplus.org/），虽然它不是一个完整的集成开发环境（integrated development environment，IDE），但它的速度非常快。当然，读者也可以选择 Microsoft 的 VS Code（https://code.visualstudio.com/）、PyCharm（https://www.jetbrains.com/pycharm/）或任何其他喜欢的工具。

2.5.2 安装 Git

请确保已安装 Git，这是一款用于版本控制的必备工具，帮助管理和跟踪代码变更，也便于团队协作。此外，Git 对于克隆代码仓库至关重要，比如我们将用到本书的代码仓库。

请访问官方 Git 网站（https://git-scm.com/book/en/v2/GettingStarted-Installing-Git）下载适合个人系统的安装包。

本书提供的所有示例代码片段及完整项目代码，均托管在这个 GitHub 仓库：https://github.com/PacktPublishing/Building-Data-Driven-Applications-with-LlamaIndex。

因此，如果想将项目文件下载到本地，在安装 Git 后，只需按以下步骤操作：

（1）进入目标目录，打开命令提示符或终端窗口，使用 cd 命令切换到想要存放项目的目录。

```
cd path/to/your/directory
```

（2）克隆仓库：运行以下命令克隆 GitHub 仓库：

```
git clone https://github.com/PacktPublishing/Building-DataDriven-Applications-with-LlamaIndex
```

此操作会将项目的副本下载到本地计算机。

（3）进入项目目录，使用如下命令进入新创建的项目文件夹。

```
cd Building-Data-Driven-Applications-with-LlamaIndex
```

在项目实践过程中，读者可以选择以下两种方式。

- 自己编写代码，之后与仓库中的代码进行对比。
- 或者可以直接查看仓库中的代码文件，以便更好地理解代码结构。

当读者正确地完成了上述步骤后，列出当前文件夹的内容，会看到多个名为 chX 的子文件夹（其中 X 是章编号），以及一个名为 PITS_APP 的独立子文件夹。这些章文件夹内含各章对应的示例源文件，而 PITS_APP 文件夹则包含主项目的源代码。

2.5.3　安装 LlamaIndex

在终端中运行以下命令安装 LlamaIndex 及其集成模块：

```
pip install llama-index
```

此命令会安装 LlamaIndex 包，该包不仅包含 LlamaIndex 的核心组件，还附带了一系列实用的集成模块。为了确保部署效率，读者也可以选择仅安装最基本的核心组件和必要的集成。不过，对于本书的学习目的来说，上述推荐的安装方式已经足够了。

> **注意**
>
> 如果读者当前使用的 LlamaIndex 版本低于 v0.10，建议在新的虚拟环境中重新安装以避免冲突。详细迁移说明请参阅：`https://pretty-sodium-5e0.notion.site/v0-10-0-Migration-Guide-6ede431dcb8841b09ea171e7f133bd77`。

完成安装后，读者就可以导入 LlamaIndex 并开始使用了。

2.5.4　注册 OpenAI 获取 API 密钥

LlamaIndex 默认集成 OpenAI 提供的 GPT 模型[①]，使用时需要配置 API 密钥用于认证。

① 译者注，如果读者使用 OpenAI 密钥不方便，也可以使用 DeepSeek 密钥，申请地址是 https://platform.deepseek.com/usage。

请前往 `https://platform.openai.com` 完成注册并登录，读者可以创建一个新的 API 密钥。请务必妥善保管好密钥。

LlamaIndex 与 OpenAI 的模型交互时均会用到这个密钥。出于安全性考虑，建议将密钥存储在本地计算机的环境变量中。

1. Windows 用户配置

在 Windows 环境中，读者可以通过以下步骤实现这一点。

（1）打开环境变量：打开“开始”菜单并搜索“环境变量”，或者右键单击“此电脑”或“我的电脑”并选择“属性”。

（2）然后单击 Advanced system settings 选项，单击 Advanced 选项卡中的 Environment Variables...按钮，参见图 2.4。

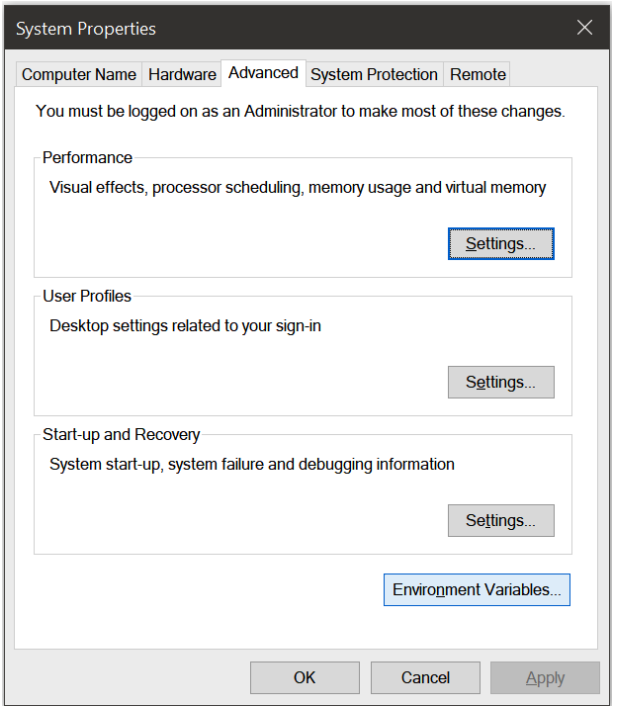

图 2.4　编辑 Windows 环境变量

（3）创建新的环境变量：在 Environment Variables 窗口中，User variables 部分下，单击 New 按钮。

（4）输入变量详情：Variable name 输入 `OPENAI_API_KEY`。Variable value 粘贴从 OpenAI 获得的 API 密钥，如图 2.5 所示。

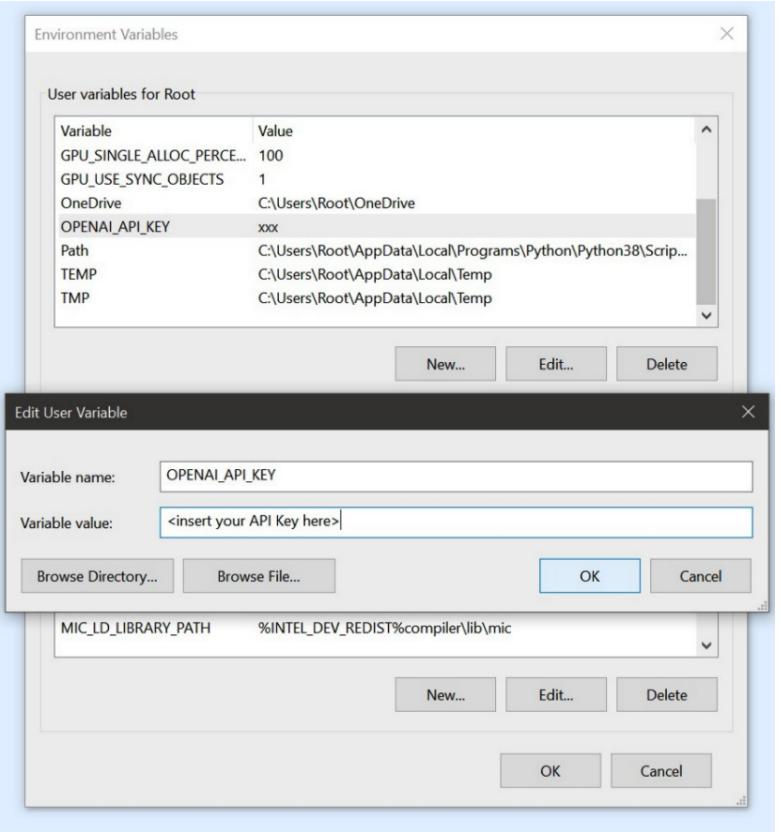

图 2.5　创建 OPENAI_API_KEY 环境变量

（5）确认并应用：单击 OK 按钮关闭所有的对话框。需要重启计算机才能使更改生效。

（6）验证环境变量：为确保已正确设置，打开新的命令提示符并运行以下命令。

```
echo %OPENAI_API_KEY%
```

该命令应显示读者刚刚存储的 API 密钥。

2. Linux/Mac 用户配置

在 Linux/Mac 上，读者可以通过以下步骤完成 OpenAI API 密钥的注册。

（1）在终端中运行以下命令，将<yourkey>替换为读者的 API 密钥。

```
echo "export OPENAI_API_KEY='yourkey'" >> ~/.zshrc
```

（2）更新 shell 以包含新的变量。

```
source ~/.zshrc
```

（3）使用以下命令，确保已经设置了环境变量。

```
echo $OPENAI_API_KEY
```

OpenAI API 密钥现在存储在环境变量中，这样密钥可以轻松地被 LlamaIndex 访问，而不会在代码或系统中暴露。

> **注意**
>
> 尽管 OpenAI 为其 GPT 模型提供了通过 API 访问的免费试用选项，但提供的免费信用额度是有限的。目前，用户可以获得最多 5 美元的免费信用额度，并且这些额度会在 3 个月内过期。这应该足以满足项目实验和阅读本书的目的。不过，如果读者希望认真构建基于大语言模型的应用程序，则需要在 OpenAI 平台上注册付费账户。另一种是使用其他 AI 模型与 LlamaIndex 配合工作。我们将在第 10 章详细讨论如何自定义 AI 模型。

至此，后端环境配置完毕。接下来将继续配置完整的技术栈。

2.5.5 Streamlit 快速构建和部署应用的理想工具

在开始构建像 PITS 这样的应用程序之前，我们需要一个平台构建和运行它们。这时 Streamlit 就派上了用场。Streamlit 是一个出色的开源 Python 库，它让创建和部署 Web 应用

程序和仪表板变得异常简单。

只需要几行 Python 代码，读者就可以构建完整的 Web 界面并及时查看效果。Streamlit 最大的优势是，Streamlit 应用程序几乎可以在任何地方部署——在服务器上、在 Heroku 等平台，甚至可直接从 GitHub 处部署。

Streamlit 的优势在于，它让开发者能够专注于应用逻辑的实现，比如使用 LlamaIndex 创建 PITS，而不是在复杂的 Web 开发上纠结。对于 AI 实验，它是完美的。

我们将主要使用 Streamlit 构建上传学习指南和与 PITS 导师互动的界面。在接下来的章节中，我们会使用 Streamlit 进行本地运行和测试应用程序。然而，在第 9 章中，我们还将进一步探讨如何用 Streamlit Share 或任何其他托管服务部署我们的应用程序。

Streamlit 拥有丰富而灵活的功能，如数据框、图表和小部件，但现在不用担心学习所有这些组件。在构建功能的过程中，我们将解释相关部分，以便你一路掌握 Streamlit 技能。

2.5.6　安装 Streamlit

最后安装 Streamlit 库。

```
pip install streamlit
```

至此，核心环境配置完成。我们有了后端工具（LlamaIndex）、前端层（Streamlit）和目标（PITS）。现在是时候做最后的润色了。

2.5.7　完成环境配置

因为我们的项目需要支持 PDF 和 DOCX 文档的处理，所以还需要安装另外两个库。

```
pip install pypdf
pip install docx2txt
```

至此，我们的开发环境已经为使用 LlamaIndex 做好准备。

现在，让我们回顾一下目前所具备的资源。

- Python 3.11。
- Git 版本控制系统。
- LlamaIndex 库。
- OpenAI 账户和 API 密钥。
- Streamlit，用于快速搭建应用程序界面的工具。
- 数据处理：PyPDF 和 DOC2Txt。

2.5.8 最终检查

为了验证开发环境配置成功，可依次执行以下命令检查相关依赖项。

```
python --version
git --version
pip show llama-index
echo %OPENAI_API_KEY%
pip show streamlit
pip show pypdf
pip show docx2txt
```

如无异常，即可运行 ch2 子文件夹中名为 `sample1.py` 的文件进行验证。

```
python sample1.py
```

执行上述命令，读者可看到位于 ch2/files 子文件夹下两个示例文档的简要概述。

如果读者遇到任何问题，请务必返回并仔细检查之前的步骤，以确保没有遗漏任何关键环节。这样做有助于避免后续工作中可能出现的问题。

至此，开发环境已配置完成。我们具备了利用 LlamaIndex 构建个性化智能辅导系统 PITS 所需的全部基础工具，这让我迫不及待地想动手实践了。

在接下来的章节中，我们将动手实践第一个 LlamaIndex 程序。从这里开始，我们将体

验到真正的乐趣：探索数据导入、索引构建、查询执行等一系列功能。每一步都配有清晰的代码示例和概念讲解。旨在帮助读者系统掌握 LlamaIndex 的核心功能与使用方法。在掌握了这些核心技能后，读者可进一步扩展 PITS 系统的功能，迈出构建智能应用的第一步。

在此之前，让我们先了解 LlamaIndex 框架代码仓库的整体结构。

2.6　熟悉 LlamaIndex 代码仓库的组织结构

为便于后续查阅和开发，读者有必要了解 LlamaIndex 官方代码仓库的整体结构。熟悉其组织方式将有助于提升项目构建与调试效率。官方代码仓库位于 `https://github.com/run-llama/llama_index`。

自 0.10 版本起，LlamaIndex 的代码结构进行了模块化重构，旨在减少不必要的依赖加载，提高运行效率，同时增强代码的可读性和开发人员的用户体验。

图 2.6 描述了 LlamaIndex 代码结构的主要组成部分。

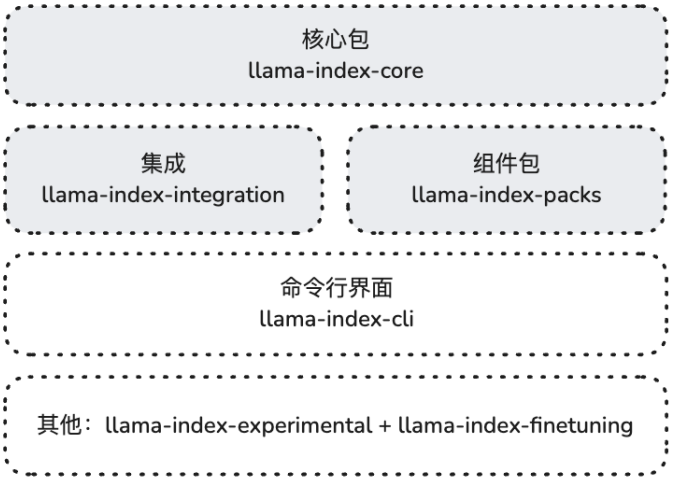

图 2.6　LlamaIndex Github 代码仓库结构

llama-index-core 文件夹包含了 LlamaIndex 的核心组件，提供了安装基本框架的功能，并支持开发者根据需求选择性地添加超过 300 个集成包和多种 Llama-packs，以实现定制化功能。

llama-index-integrations 文件夹包含各种拓展包，用以扩展核心框架的功能。这些拓展包允许开发人员选择特定元素（如自定义大语言模型、数据读取器、嵌入模型和向量存储提供者）定制他们的构建，以最好地满足其应用的需求。我们将从第 4 章开始探讨其中的部分集成。

llama-index-packs 文件夹包含超过 50 个 Llama 包，由 LlamaIndex 开发者社区持续开发和完善。这些包作为预配置的模板，旨在帮助用户快速启动应用。更多详情请参见第 9 章。

llama-index-cli 文件夹用于实现 LlamaIndex 命令行接口，相关内容也将在第 9 章中提及。

图 2.6 中的最后一部分“其他：”包含两个文件夹，目前它包含微调抽象和一些实验性功能，这些内容不在本书的讨论范围内。

注意

llama-index-integrations 和 llama-index-packs 下的子文件夹分别代表各个独立的软件包。文件夹名称即为对应的 PyPI 包名称。例如，llama-index-integrations/llms/llama-index-llms-mistralai 文件夹对应于 llama-index-llms-mistralai 的 PyPI 包。

根据这个例子，使用如下命令导入和使用 mistralai 包。

```
from llama_index.llms.mistralai import MistralAI
```

可通过以下命令安装相应的 PyPI 包。

```
pip install llama-index.llms.mistralai
```

读者无需担心遗漏书中包含的示例所需的重要包，因为每章的技术需求部分都已列出了所需要的包。

2.7　本章小结

在本章中，我们介绍了 LlamaIndex，一种用于将大语言模型与外部数据源连接的框架。我们探讨了该框架如何增强模型的知识获取能力，使其回答更加贴近现实语境。

我们还对比了 LlamaIndex 与模型微调方案的不同，指出其在数据更新效率与个性化支持方面的显著优势。此外，我们引入“渐进式复杂度揭示”的理念，即 LlamaIndex 初始使用门槛较低，但在功能拓展时具备良好的可拓展性和灵活性。

本章还引入了实践项目 PITS，并详细讲解了开发环境配置的各项工具（如 Python、Git 和 Streamlit）的安装与配置过程，以及 OpenAI API 密钥的获取方法。随着开发环境的成功搭建，我们已为后续的 LlamaIndex 应用开发做好了准备。

现在我们可以继续深入探索，学习更多关于 LlamaIndex 框架的内部工作原理和技术细节。期待在第 3 章中继续与你一同深入探索。