初窥LLMOps，助力大模型落地
章节介绍

LLM到AI Agent的技术演进

- 详细介绍Agent的基本概念、组成部分，以及Agent是如何弥补LLM的不足。
- 在ChatGPT对话中模拟一个Agent的运行流程，实现一个拥有查询ip、实时搜索、数学计算能力的“Agent”。
- 探讨开发、测试、部署、更新AI Agent的疑难点，引出后端即服务和LLMOps的概念。

初窥LLMOps及课程项目拆解

- 介绍什么是LLMOps，以及为什么需要使用LLMOps，使用LLMOps对企业构建AI应用能带来什么价值。
- 展示目前市面上类LLMOps平台，涵盖Coze、Dify、语聚、智谱清言等，并对其功能进行演示。
- 基于市面上的LLMOps平台拆解课程中开发的LLMOps项目，并展示相关的项目文档与源码。

从LLM大模型到AI Agent的技术演进

LLM/RAG/Agent的技术路线

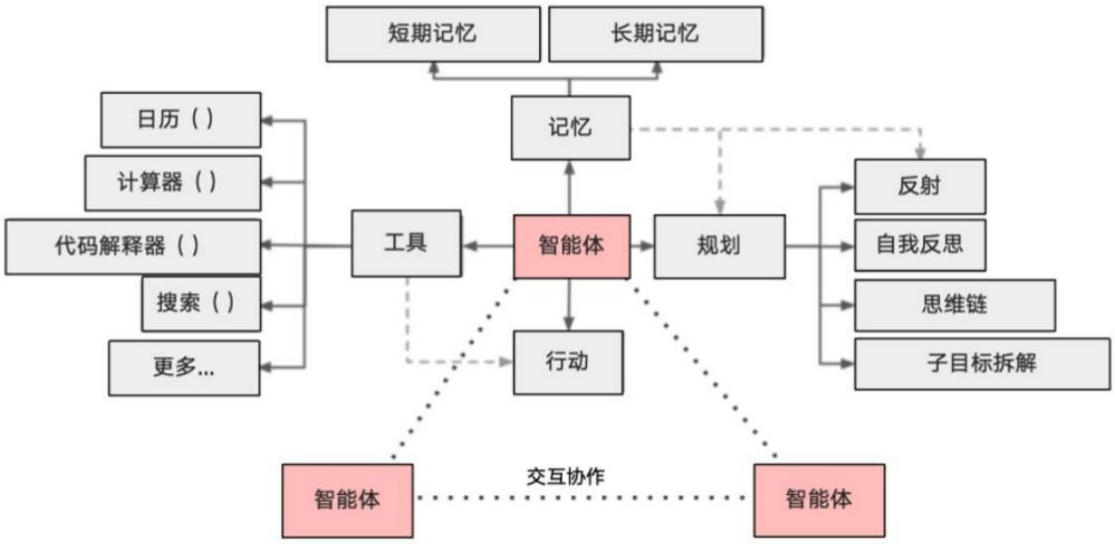
- LLM/RAG/Agent已经成为人工智能领域进步的关键技术，理解这三者的概念与关系是做好面向AI编程开发的基础。

	大模型(LLM)	索引增强搜索(RAG)	智能体(Agent)
定义	大型语言模型（LLM），如GPT系列、BERT等，是利用大量文本数据训练的模型，能够生成连贯的文本、理解语言、回答问题等。	检索增强生成技术结合了传统的信息检索技术和最新的生成式模型。它先从一个大型的知识库中检索出与查询最相关的信息，然后基于这些信息生成回答。	智能体是指具有一定智能的程序或设备，能够感知环境并根据感知结果做出响应或决策的实体。它们可以是简单的软件程序或复杂的机器人。
作用	LLM作为基础技术，提供了强大的语言理解和生成能力，是构建复杂人工智能系统的基石。	RAG可以视为在LLM基础上的扩展或应用，利用LLM的生成能力和外部知识库的丰富信息来提供更准确、信息丰富的输出。	智能体可以利用LLM进行自然语言处理，通过RAG技术获得和利用知识，以在更广泛的环境中做出决策和执行任务。它们通常位于应用层级，是对LLM和RAG技术在特定环境下的集成和应用。

AI Agent的定义与技术架构

- OpenAI 将AI Agent定义为，以大语言模型为大脑驱动，具备自主感知能力、规划、记忆和使用工具的能力，能够自动执行完成复杂任务的系统。

AI Agent的定义与技术架构



手动模拟一个简单的Agent

- Agent无论传递了多少的工具、记忆、规划、行动等，本质上都是统一编码到Prompt中并传递给大语言模型，然后程序根据输出的内容执行不同的操作，反复执行，直到完成。
- 任何一个Agent都可以在ChatGPT或者大模型的对话界面进行模拟，并且实现类似的效果。
- 实现一个集ip地址查询、数学表达式计算、网络搜索的简单Agent应用。

手动模拟Agent流程图

课程总结

- 学习基础LLM是如何演化出RAG与Agent的，这三者之间的关系是什么样的。
- 分享了Agent是如何增强LLM的能力的，讲解了基于LLM的Agent架构图，以及Agent的运行流程。
- 在ChatGPT对话界面，使用手动对话+手动录入工具结果的方式模拟了一个Agent的运行流程。

初识LLMOps，为什么需要LLMOps

低门槛创建AI Agent应用?

- 一个AI Agent应用涵盖了LLM、记忆、工具、Prompt、规划、知识库、执行者等模块，但每个应用的流程又比较接近，对开发者和非开发者都不友好。
- 有没有一个平台，能在可视化界面通过鼠标拖拽对应的模块+提示词等内容，就可以让无编程基础的人也可以快速创建一个AI Agent应用并调试部署?

LLMOps的定义与目标

- LLMOps是一个基于LLM的应用程序的生命周期管理平台或者工具，涵盖了LLM应用的开发、部署、配置、运维。
- LLMOps的旨在简化和优化LLM应用程序的各个环节，以确保LLM应用高效、可靠和安全地运行。
- LLMOps对使用者友好，极大降低了企业创建AI Agent应用的成本，把复杂的部分留给了LLMOps开发者。

AI应用开发各环节差异

步骤	未使用LLMOps平台	使用LLMOps平台	时间差异
开发应用前&后端	集成和封装LLM能力，花费较多时间开发前端应用	直接使用LLMOps的后端服务，可基于开放API、WebApp直接开发	-80%
提示工程	仅能通过API或Playground进行	Prompt可视化编排，所见即所得完成调试	-25%
数据准备与嵌入	编写代码实现长文本数据处理、嵌入	在LLMOps平台上传文本、文件	-80%
应用日志与分析	编写代码查看日志，访问数据库查看	平台提供实时日志与分析	-70%
AI插件开发与集成	编写代码创建、集成AI插件	平台提供可视化工具创建、快速集成自定义插件能力	-50%
AI工作流开发	编写代码完成每一个工作流	可视化编排工作流，所见即所得调试	-80%

两种模式开发AI应用流程对比

- **不使用LLMOps**：整理需求、Prompt编写、部署LLM、对接LLM接口、处理应用数据、记录日志、应用工具开发、AI工作流开发、前后端交互开发、前后端部署。

- **使用LLMOps**：需求整理、Prompt编写、上传数据、勾选关联插件、可视化编排工作流、选择LLM模型、发布。

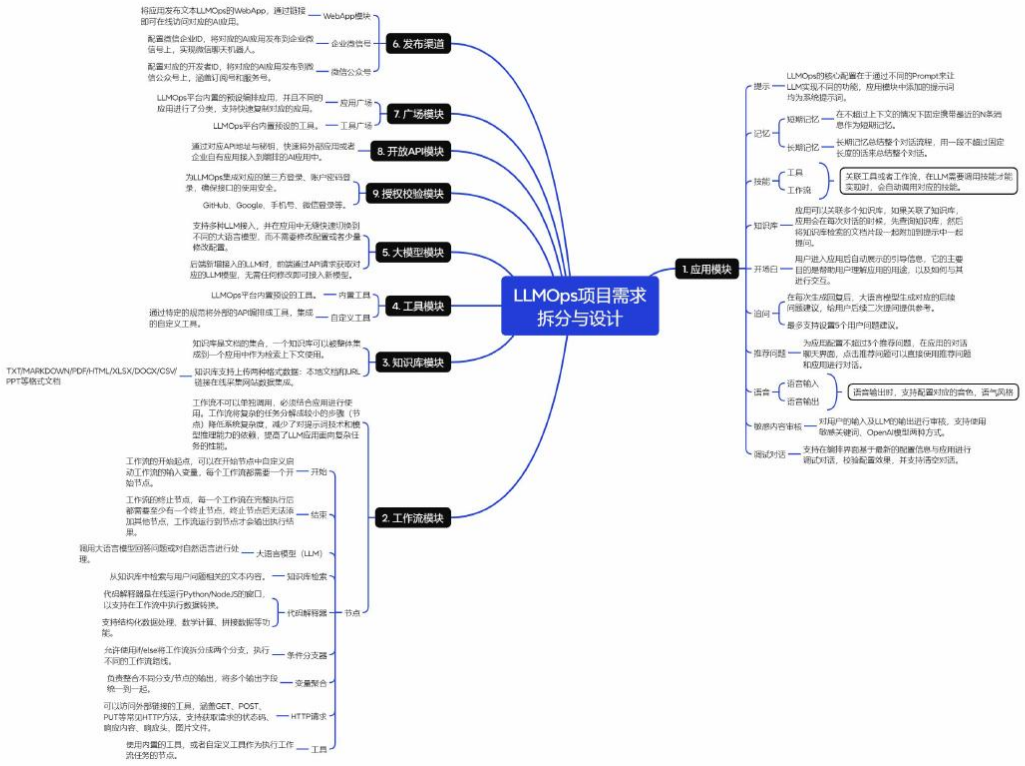
课程总结

- 学习了LLMOps的定义，能为企业带来什么价值，为什么需要LLMOps。

- 了解在使用和不使用LLMOps进行AI应用开发时各个环节的差异与时间对比，以及LLMOps带来的便利性。

Dify LLMOps应用开发平台功能演示

LLMOps项目需求拆分与设计



课程学习目标与解决的问题展示

课程学习目标

- 在开发LLMOps项目的过程中，精通LangChain框架的使用及底层原理，掌握AI应用开发必备的技能。
- 能独立提炼对应的AI应用开发需求并对其进行拆分，完成开发、测试、部署与运维全流程。
- 读懂LangChain绝大部分组件的代码，初步具备跨语言开发AI应用框架的能力。

解决的AI应用开发问题（一）

- Prompt编写：通过对LLM的测评，了解不同的Prompt编写技巧，以适配不同的LLM。
- 多LLM接口对齐：解决不同LLM接口无缝对接到同一个应用中，并实现少量修改配置甚至不修改可正常运行。
- 多LLM消耗统计：解决统计不同LLM接口输入与输出消耗，精准计算接口消耗成本。

解决的AI应用开发问题（二）

- LLM实现短期/长期记忆：通过附加最近的N条消息，与长文总结，在上下文长度限制内，实现LLM精准短期记忆，模糊长期记忆。
- 需求转Agent：掌握将复杂需求转换成Agent，并完成开发、配置、测试与部署的整个流程。
- RAG优化：解决不同成本下的RAG优化，涵盖本地搜索、向量搜索、搜索重排。

解决的AI应用开发问题（三）

- 不同场景下的流式响应：解决在不同场景下LLM回复时间过长的问题，涵盖流式响应、流式响应转正常响应技巧。
- AI工作流编排：掌握将一个复杂任务拆分成多个小任务，并完成工作流的合理编排与开发，使用不同的LLM解决不同的任务，提升性能降低成本。

课程源码与项目资料

章节总结