1. 大模型简介

**一、大模型的概念**

**大语言模型（LLM，Large Language Model），也称大型语言模型，是一种旨在理解和生成人类语言的人工智能模型**。

二、常见的LLM模型

国外知名的 LLM 有：GPT-3.5、GPT-4、PaLM、Claude 和 LLaMA 等。

国内知名的 LLM 有：文心一言、讯飞星火、通义千问、ChatGLM、百川等。

三、LLM的发展历程

1、20世纪90年代：语言建模研究开始。主要集中在采用**统计学习方法进行**词汇预测，通过分析前面的词汇来预测下一个词汇。但在理解复杂语言规则方面存在一定局限性。

2、2003年：**Bengio** 在《A Neural Probabilistic Language Model》中，首次将深度学习的思想融入到语言模型中。

3、2018年左右：**Transformer 架构的神经网络模型**开始崭露头角。通过大量文本数据训练，极大地提升了模型在各种自然语言处理任务上的表现。

4、2022年：4月[PaLM/Gemini 系列](https://datawhalechina.github.io/llm-universe/" \l "/C1/1.%E5%A4%A7%E8%AF%AD%E8%A8%80%E6%A8%A1%E5%9E%8B LLM %E7%90%86%E8%AE%BA%E7%AE%80%E4%BB%8B?id=_13113-palmgemini-%e7%b3%bb%e5%88%97)发布

5、2023年：

3 月，文心一言开启邀测

3 月，[PaLM/Gemini 系列](https://datawhalechina.github.io/llm-universe/" \l "/C1/1.%E5%A4%A7%E8%AF%AD%E8%A8%80%E6%A8%A1%E5%9E%8B LLM %E7%90%86%E8%AE%BA%E7%AE%80%E4%BB%8B?id=_13113-palmgemini-%e7%b3%bb%e5%88%97)公开了 API

3 月 15 日，Claude发布

4 月，通义千问正式发布

5 月，Google 发布了PaLM 2

5 月，讯飞星火认知大模型发布

7 月 11 日，Claude更新至Claude-2

11 月 7 日， OpenAI 推出了GPT-4 Turbo

6、2024年：

3 月 4 日，Claude更新至 Claude-3

5 月 14 日，GPT-4o正式发布

6 月 6 日，Qwen2正式开源

1. LLM的能力与特点

（一）三个LLM典型的涌现能力：

（1）上下文学习：根据提供自然语言指令或多个任务示例中，通过理解上下文并生成相应输出的方式来执行任务，而无需额外的训练或参数更新。

（2）指令遵循：LLM 能够根据任务指令执行任务，而无需事先见过具体示例，展示了其强大的泛化能力。

（3）逐步推理：LLM 通过采用思维链（CoT, Chain of Thought）推理策略，利用包含中间推理步骤的提示机制来解决这些任务，从而得出最终答案。

1. 作为基座模型支持多元应用的能力

多个应用可以只依赖于一个或少数几个大模型进行统一建设。

1. 支持对话作为统一入口的能力
2. LLM的特点
3. 巨大的规模：可达数十亿甚至数千亿个参数。能捕捉更多的语言知识和复杂的语法结构。
4. 预训练和微调：在大规模文本数据上进行预训练，学习通用的语言表示和知识。然后通过微调适应特定任务。
5. 上下文感知：能够理解和生成依赖于前文的文本内容。
6. 多语言与模态支持：可理解多种语言和生成不同媒体类型的内容。
7. 伦理和风险问题：引发了包括生成有害内容、隐私问题、认知偏差等。
8. 高计算资源需求：参数规模庞大，需要大量的计算资源进行训练和推理。通常需要使用高性能的 GPU 或 TPU 集群来实现。
9. 检索增强生成（RAG, Retrieval-Augmented Generation）
10. RAG基本概念

整合了从庞大知识库中检索到的相关信息，并以此为基础，指导大型语言模型生成更为精准的答案。

1. LLM面临的主要问题

（1）信息偏差/幻觉： LLM有时会会产生与客观事实不符的信息，导致用户接收到的信息不准确。RAG 通过检索数据源，辅助模型生成过程，确保输出内容的精确性和可信度，减少信息偏差。

（2）知识更新滞后性： LLM 基于静态的数据集训练，这可能导致模型的知识更新滞后，无法及时反映最新的信息动态。RAG 通过实时检索最新数据，保持内容的时效性，确保信息的持续更新和准确性。

（3）内容不可追溯： LLM 生成的内容往往缺乏明确的信息来源，影响内容的可信度。RAG 将生成内容与检索到的原始资料建立链接，增强了内容的可追溯性，从而提升了用户对生成内容的信任度。

（4）领域专业知识能力欠缺： LLM 在处理特定领域的专业知识时，效果可能不太理想，这可能会影响到其在相关领域的回答质量。RAG 通过检索特定领域的相关文档，为模型提供丰富的上下文信息，从而提升了在专业领域内的问题回答质量和深度。

（5）推理能力限制：LLM 在处理复杂问题时可能缺乏必要的推理能力，影响了其对问题的回答质量。RAG 结合检索到的信息和模型的生成能力，通过提供额外的背景知识和数据支持，增强了模型的推理和理解能力。

（6）应用场景适应性受限： LLM 需在多样化的应用场景中保持高效和准确，但单一模型可能难以全面适应所有场景。RAG 使得 LLM 能够通过检索对应应用场景数据的方式，灵活适应问答系统、推荐系统等多种应用场景。

1. 长文本处理能力较弱： LLM 在理解和生成长篇内容时受限于有限的上下文窗口，且必须按顺序处理内容，输入越长，速度越慢。RAG 通过检索和整合长文本信息，强化了模型对长上下文的理解和生成，有效突破了输入长度的限制，同时降低了调用成本，并提升了整体的处理效率。

（三）RAG的工作流程

1、数据处理阶段

（1）对原始数据进行清洗和处理。

（2）将处理后的数据转化为检索模型可以使用的格式。

（3）将处理后的数据存储在对应的数据库中。

2、检索阶段

（1）将用户的问题输入到检索系统中，从数据库中检索相关信息。

3、增强阶段

（1）对检索到的信息进行处理和增强，以便生成模型可以更好地理解和使用。

4、生成阶段

（1）将增强后的信息输入到生成模型中，生成模型根据这些信息生成答案。

（四）RAG VS Finetune（微调）

| 特征比较 | RAG | 微调 |
| --- | --- | --- |
| 知识更新 | 直接更新检索知识库，无需重新训练。信息更新成本低，适合动态变化的数据。 | 通常需要重新训练来保持知识和数据的更新。更新成本高，适合静态数据。 |
| 外部知识 | 擅长利用外部资源，特别适合处理文档或其他结构化/非结构化数据库。 | 将外部知识学习到 LLM 内部。 |
| 数据处理 | 对数据的处理和操作要求极低。 | 依赖于构建高质量的数据集，有限的数据集可能无法显著提高性能。 |
| 模型定制 | 侧重于信息检索和融合外部知识，但可能无法充分定制模型行为或写作风格。 | 可以根据特定风格或术语调整 LLM 行为、写作风格或特定领域知识。 |
| 可解释性 | 可以追溯到具体的数据来源，有较好的可解释性和可追踪性。 | 黑盒子，可解释性相对较低。 |
| 计算资源 | 需要额外的资源来支持检索机制和数据库的维护。 | 依赖高质量的训练数据集和微调目标，对计算资源的要求较高。 |
| 推理延迟 | 增加了检索步骤的耗时 | 单纯 LLM 生成的耗时 |
| 降低幻觉 | 通过检索到的真实信息生成回答，降低了产生幻觉的概率。 | 模型学习特定领域的数据有助于减少幻觉，但面对未见过的输入时仍可能出现幻觉。 |
| 伦理隐私 | 检索和使用外部数据可能引发伦理和隐私方面的问题。 | 训练数据中的敏感信息需要妥善处理，以防泄露。 |

1. LangChain
2. 什么是LangChain

LangChain 框架是一个开源工具，充分利用了大型语言模型的强大能力，以便开发各种下游应用。它的目标是为各种大型语言模型应用提供通用接口，从而简化应用程序的开发流程。

1. LangChain的核心组件

1、模型输入/输出（Model I/O）：与语言模型交互的接口。

2、数据连接（Data connection）：与特定应用程序的数据进行交互的接口。

3、链（Chains）：将组件组合实现端到端应用。比如搭建检索问答链来完成检索问答。

4、记忆（Memory）：用于链的多次运行之间持久化应用程序状态。

5、代理（Agents）：扩展模型的推理能力。用于复杂的应用的调用序列。

6、回调（Callbacks）：扩展模型的推理能力。用于复杂的应用的调用序列。

1. 开发LLM应用的整体流程
2. 确定目标：

1、拟定应用场景、目标人群、核心价值。

2、设定最小化目标，构建一个 MVP（最小可行性产品）。

（二）设计功能：

1、确定应用的核心功能。

2、设计核心功能的上下游功能。

（三）搭建整体架构：

1、构建特定数据库 + Prompt + 通用大模型的架构。

2、推荐基于 LangChain 框架进行开发。

（四）搭建数据库：

1、使用向量数据库（如Chroma）。

2、收集数据并进行预处理（数据格式转化和数据清洗）。

3、切片、向量化构建出个性化数据库。

（五）Prompt Engineering：

1、明确 Prompt 设计的一般原则及技巧。

2、构建小型验证集。

3、基于小型验证集设计满足基本要求、具备基本能力的 Prompt。

1. 验证迭代：

1、发现 Bad Case。

2、改进 Prompt Engineering 来提升系统效果、应对边界情况。

3、进行实际业务测试，探讨边界情况，找到 Bad Case。

4、不断迭代优化Prompt，直到达到一个较为稳定、可以基本实现目标的 Prompt 版本。

1. 前后端搭建：

1、搭建前后端，设计产品页面。

2、采用 Gradio 和 Streamlit，快速搭建可视化页面，实现 Demo 上线。

1. 体验优化。

1、上线应用，进行长期的用户体验跟踪。

2、记录 Bad Case 与用户负反馈，针对性进行优化。