Karpathy DeepDive LLM

2024年1月1日 2 分钟阅读 Peng Tan

#资源 #链接

> *Karpathy DeepDive LLM视频课程*

Deep Dive into Large Language Models

https://www.youtube.com/watch?v=zjkBMFhNj_g

在 YouTube 视频脚本《Deep Dive into LLMs like ChatGPT》中，主讲人 Andrej Karpathy **按照构建和理解大型语言模型的完整流程**展开讲述，他试图从一个高屋建瓴的角度，向普通观众介绍 LLM 的内部工作机制及其能力与局限性。其讲述方式大致遵循以下顺序：

> **导论与核心问题提出**：首先，Karpathy 强调了 LLM（如 ChatGPT）的魔力与强大，同时也指出了其不足和需要注意的“锋利边缘”。他提出了听众可能关心的核心问题，例如文本框后发生了什么、模型如何生成词语、以及我们究竟在与什么对话。
>
> **构建 LLM 的流程**：他随后开始详细介绍**构建一个类似 ChatGPT 的 LLM 的完整流水线 (pipeline)**，并强调这将是一个**顺序排列的多个阶段**。
>
> **预训练阶段 (Pre-training Stage)**：这是第一个阶段，重点在于**知识的获取**。
>
> **下载和处理互联网数据**：他介绍了获取大量高质量、多样性互联网文本数据的过程，并推荐了 Hugging Face 的 FineWeb 数据集作为参考。
>
> **训练目标：预测下一个 token**：他解释了预训练的核心任务是**预测序列中的下一个 token**，并提到了上下文窗口大小的限制。

目录

文章信息

字数

阅读时间

发布时间

更新时间

标签

#资源 #链接

训练数据规模：他用 GPT-2 的训练数据量（约 1000 亿 tokens）和 FineWeb 的数据量（15 万亿 tokens）进行了对比，展示了数据规模的演进。他还分享了自己复现 GPT-2 的经历和训练成本。

模型发布 (Model Release)：他解释了**发布一个基础模型**需要**模型代码（描述神经网络操作序列的 Python 代码）和模型参数（神经网络中数十亿个"旋钮"的正确设置）**。他用 GPT-2 和更现代的 Llama 3 作为例子进行了说明。他还演示了如何与一个**基础模型**（Llama 3）进行交互，强调其本质是一个"昂贵的自动补全"工具，尚未成为一个助手。

指令微调与助手模型 (Instruct Model)：他解释了如何通过在**对话数据集**上进行**持续训练 (continue training)** 将基础模型转化为可以回答问题的**助手模型**。

对话数据的格式和 tokenization：他介绍了如何将用户和助手之间的对话编码成 **token 序列**。

InstructGPT 论文：他重点介绍了 OpenAI 的 InstructGPT 论文，这是首次公开讨论如何通过在对话数据上微调语言模型以使其更像助手。

人工标注数据：他详细介绍了**人工标注员 (human labelers)** 在创建高质量对话数据中的作用，包括他们如何根据**标注指南 (labeling instructions)** 编写理想的助手回复。他还提到了 Open Assistant 等开源项目也在尝试构建类似的对话数据集。

LLM 心理学 (LLM Psychology)：他探讨了 LLM 训练过程中出现的**认知效应 (emergent cognitive effects)**，特别是**幻觉 (hallucinations)** 问题。他通过例子解释了幻觉的可能来源，并介绍了 Meta 如何通过让模型学习"不知道"来减轻幻觉。

工具的使用 (Use of Tools)：他讲解了如何通过**引入特殊 token 和训练数据**，使 LLM 能够使用外部**工具**，例如**网页搜索**，以获取更准确和最新的信息，从而减少幻觉。他区分了**模型参数中的知识（模糊的回忆）和上下文窗口中的知识（工作记忆）**。

模型身份 (Model Identity)：他解释了模型如何"认为"自己是谁（例如 ChatGPT by OpenAI），以及开发者如何通过硬编码或系统消息来覆盖这种"自我认知"。

强化学习 (Reinforcement Learning, RL)：他将 RL 视为**第三个主要的训练阶段**，通常在预训练和监督微调之后进行。他用**学生学习**的过程类比解释了这三个阶段：预训练是基础知识学习，监督微调是学习例题，而强化学习是做练习题。他强调 RL 对于**提升模型的推理能力**至关重要，并介绍了 DeepSeek 的 R1 论文作为成功应用 RL 的案例。他还对比了经过 RL 训练的"思考模型 (thinking models)"与主要基于监督微调的模型，并介绍了如何在不同的平台（DeepSeek, ChatGPT, Gemini）上体验思考

模型。最后，他用 AlphaGo 的例子说明了 RL 的潜力，即模型可以通过自我对弈发现超越人类的策略。他还讨论了 RLF (Reinforcement Learning from Feedback)，认为其更像是一种微调而非真正的 RL。

总结与未来展望：最后，Karpathy 总结了 LLM 的三个主要训练阶段，并讨论了它们的局限性（例如幻觉，“瑞士奶酪”能力）以及未来的发展趋势（例如多模态、更长的上下文窗口）。他还推荐了一些跟踪 LLM 最新进展的资源和查找模型的方法。最后，他再次回到最初的问题，用整个讲述的流程来解释当我们与 ChatGPT 交互时究竟发生了什么，并强调 LLM 本质上是对人工标注员行为的一种模拟。他再次强调了 LLM 作为工具的价值和使用时的注意事项。

此视频的文字稿详细介绍了大型语言模型（LLM）如ChatGPT的构建和运作原理。它首先阐述了预训练阶段，模型通过学习海量互联网文本来获取广泛知识，如同阅读大量书籍构建认知基础。接着，视频深入探讨了如何将基础模型转化为有用的助手，即监督微调阶段，通过人类标注的对话数据，模型学习模仿人类的对话模式和指令响应，如同学生学习例题。最后，文字稿着重讲解了强化学习阶段，特别强调了如何利用强化学习与人类反馈（RLHF） 来优化模型的行为，使其更符合人类偏好，如同学生通过练习题和反馈来精进技能。此外，视频还探讨了LLM的能力边界和局限性，例如幻觉问题以及在简单计数和逻辑推理方面可能出现的不足，并介绍了利用工具（如代码解释器和网络搜索） 来增强模型能力的方法。总而言之，该视频旨在为普通受众提供一个关于LLM工作机制的全面而易懂的解释，揭示其“魔法”背后的技术原理和训练过程。

将大型语言模型视为新兴操作系统内核的理解

理解大型语言模型（LLMs）作为新兴操作系统内核的角色，及其对计算范式的潜在影响，是一个深刻且具有前瞻性的思考。尽管LLMs与传统操作系统内核在架构和底层机制上存在显著差异，但从更高层次的抽象来看，它们正在承担或可能承担一些类似内核的关键功能，并有望引发计算范式的重大变革。

将大型语言模型视为新兴操作系统内核的理解：

这种理解并非字面意义上的替换，而是指LLMs在以下几个方面展现出与传统操作系统内核相似的核心作用：

资源管理和抽象： 传统的操作系统内核负责管理计算机的硬件资源（CPU、内存、I/O等），并为应用程序提供统一的、抽象的接口。LLMs，特别是那些具备多模态能力和工具使用能力的模型，正在成为信息和智能的中心枢纽。它们可以**整合来自各种来源的数据和知识**，并通过自然语言接口为用户和应用程序提供对这些信息和能力的**统一访问和抽象**。例如，用户可以通

过自然语言指令要求LLM进行搜索，生成文本，分析文档，甚至控制其他工具。

任务调度与执行： 操作系统内核负责调度和执行各种计算任务。LLMs通过理解自然语言指令，可以将用户的意图转化为一系列内部操作或对外部工具的调用。这种**基于自然语言的指令执行**，可以看作是一种更高级别的任务调度和执行机制，它模糊了用户与底层计算资源的直接交互。

提供核心服务： 操作系统内核提供文件系统、网络协议栈、安全机制等核心服务。LLMs正在成为**知识服务、内容生成、智能助手、决策支持**等新型核心服务的提供者。它们通过其庞大的知识库和强大的生成能力，直接满足用户的各种信息和智能需求。

构建新的软件生态系统： 传统的操作系统内核是构建应用程序生态系统的基础。围绕LLMs，我们正在看到一个**基于自然语言交互和模型能力的全新应用生态系统**的兴起。开发者可以利用LLMs的能力，构建无需大量传统编程的智能应用程序。**通过巧妙的提示工程（Prompt Engineering）**，开发者可以引导LLM完成各种复杂的任务，这类似于在操作系统之上调用不同的API。

上下文管理与状态维护： 操作系统内核需要管理进程的上下文和系统状态。LLMs通过**上下文窗口（Context Window）**来维护对话历史和当前任务的状态，这使得它们能够进行连贯的、有记忆的交互。虽然机制不同，但目标都是为了支持持续性的操作和交互。

大型语言模型对计算范式的潜在影响：

LLMs作为新兴的"智能内核"，可能对当前的计算范式产生以下深远影响：

自然语言成为主要的交互方式： 传统的计算范式依赖于图形用户界面（GUI）或命令行界面（CLI）。LLMs的普及可能会推动**自然语言成为用户与计算系统交互的主要方式。**用户不再需要学习复杂的命令或操作流程，只需用日常语言表达意图，系统就能理解并执行。

软件开发模式的变革： 基于LLMs的应用开发将更加**侧重于定义问题、设计交互流程和优化提示。**传统的编码工作量可能会大幅减少，开发者可以更专注于业务逻辑和用户体验。**低代码甚至无代码开发**的潜力将得到极大的释放。

智能应用的普及： LLMs使得构建**具备高度智能化和自主性的应用程序**成为可能。这些应用可以理解复杂的意图，处理各种非结构化数据，进行推理和决策支持，极大地拓展了计算机的应用领域。

知识的民主化与普惠： LLMs可以作为**庞大知识库的接口**，使得用户能够以简单、直接的方式获取和利用信息。这有助于打破

信息壁垒，促进知识的普及和共享。

个性化和自适应计算： LLMs能够理解用户的偏好和上下文，提供**更加个性化和自适应的服务**。未来的计算系统可能会更加智能地响应用户的需求，提供定制化的信息和功能。

模糊人机界限： LLMs强大的自然语言理解和生成能力，使得人机对话更加自然流畅。这可能会模糊人机之间的界限，创造出更具协作性和伴随性的智能系统。

对传统操作系统的补充与挑战： LLMs并非要完全取代传统操作系统，更可能的是**在传统操作系统之上构建一个“智能层”**，提供更高级别的抽象和智能服务。然而，在某些特定领域，例如专注于信息处理和自然语言交互的应用场景，LLMs可能会承担更多核心功能，对传统操作系统构成一定的挑战。

需要注意的方面：

可靠性和安全性： LLMs仍然存在**幻觉（Hallucination）的问题，即生成不真实或不准确的信息。此外，LLMs也面临安全风险，例如越狱攻击（Jailbreak Attacks）和提示注入（Prompt Injection）**，这些都需要得到有效的解决。

计算资源消耗： 训练和运行大型LLMs需要巨大的计算资源。如何更高效地利用和部署这些模型，是未来需要关注的关键问题。

伦理和社会影响： LLMs的广泛应用引发了诸多伦理和社会问题，例如偏见、滥用、就业影响等，需要进行深入的思考和规范。

总而言之，将大型语言模型理解为新兴操作系统内核的角色，是一种富有洞察力的视角。虽然它们与传统内核的实现方式截然不同，但它们在**管理信息和智能资源、调度和执行任务、提供核心服务以及构建新的应用生态系统**等方面展现出相似的核心功能。LLMs的持续发展和普及，有望深刻地改变我们与计算机交互的方式，推动计算范式向着**更自然、更智能、更个性化**的方向发展。然而，我们也必须正视其挑战，并积极应对，以确保这项新兴技术能够为人类带来福祉。

作为技术读者，你应如何通过使用 NotebookLM 来快速掌握这些内容？

使用 NotebookLM 快速掌握这些内容，你可以采取以下步骤：

上传视频脚本到 NotebookLM：将整个 YouTube 视频的文字稿上传到 NotebookLM。

利用 NotebookLM 的摘要功能：首先，让 NotebookLM **生成整个脚本的摘要。**这将帮助你快速了解视频的主要内容和结构，抓住核心论点。

识别关键章节和主题：借助 NotebookLM 的**主题识别或章节划分功能**（如果存在），或者通过阅读摘要和标题，**识别 Karpathy 讲述的各个主要阶段和主题**，例如预训练、模型发布、指令微调、LLM 心理学、工具使用、强化学习等。

针对每个主题提出具体的技术问题：对于你感兴趣或需要深入理解的技术细节，**向 NotebookLM 提出具体的问题。**例如：

“预训练阶段的关键技术是什么？”

“上下文窗口的大小对模型性能有何影响？”

“基础模型和指令模型在架构或训练目标上有何不同？”

“人工标注员在指令微调中扮演什么角色？”

“幻觉问题是如何产生的？有哪些缓解方法？”

“模型如何利用工具进行网页搜索？”

“强化学习是如何提升模型的推理能力的？DeepSeek R1 的案例说明了什么？”

“开放权重模型和闭源模型的主要区别是什么？”

利用 NotebookLM 的问答和引用功能：NotebookLM 会根据你的问题在脚本中**查找相关信息并给出答案。**注意查看答案**引用的原始文本片段**，以便更准确地理解上下文和细节。

使用 NotebookLM 的笔记功能总结关键概念：在阅读 NotebookLM 的答案和原始文本后，**使用 NotebookLM 的笔记功能**，**总结每个阶段的关键概念、技术术语、模型名称和重要结论。**例如，记录下不同模型的参数量、训练数据量、以及它们擅长的任务。

比较和对比不同的概念和模型：如果你想比较不同的训练方法（如监督微调和强化学习）或不同的模型（如 GPT-2 和 Llama 3），可以**向 NotebookLM 提出比较性的问题**，并利用笔记功能整理对比结果。

关注技术细节和术语解释：Karpathy 在视频中会提到一些技术术语（例如 tokens, parameters, neural network, transformer）。利用 NotebookLM **针对这些术语提问**，确保你理解它们的含义。

略过或快速浏览已知内容：作为技术读者，你可能对某些基础概念已经有所了解。利用 NotebookLM 的摘要和快速浏览功能，**跳过你熟悉的部分**，将注意力集中在新的或需要深入理解

的内容上。

通过以上步骤，你可以利用 NotebookLM 快速有效地从 Andrej Karpathy 的视频脚本中提取关键的技术信息，构建对大型语言模型构建流程和核心概念的深入理解。NotebookLM 的问答和引用功能能够帮助你快速定位和理解脚本中的重要细节，而笔记功能则可以帮助你整理和巩固所学知识。

分享这篇文章

相关文章推荐

ai usa impact

ai usa impact 相关资源链接

claude mcp server

claude mcp server相关资源...

claudia

claudia 相关资源链接