deepseek复现开源项目

[huggingface/open-r1：DeepSeek-R1 的全开放复刻版](https://github.com/huggingface/open-r1)

deepseek可以做什么

[深度搜索-ai/awesome-deepseek-集成](https://github.com/deepseek-ai/awesome-deepseek-integration)

1.集成web、客户端应用

2.论文阅读器

3.数据分析

4.桌面助手

5.ide插件

6.浏览器插件

7.语音助手

8.代码生成

9.漏洞修复

…

deepseek promot

deepseek promot官方文档

[Prompt Library | DeepSeek API Docs](https://api-docs.deepseek.com/zh-cn/prompt-library/)

deepseek promot特性，promot与其他大模型的差异

[DeepSeek-R1直接把提示工程师的饭碗整没了 - 知乎](https://zhuanlan.zhihu.com/p/20697018627)

[DeepSeek R1: ”你在教我做事？“ - 知乎](https://zhuanlan.zhihu.com/p/20418009560)

(1)使用清晰、简洁且无歧义的提示。

(2)避免对DeepSeek-R1使用few-shot提示，因为这已被证明会在复杂任务中降低性能。相反，使用zero-shot或结构化提示。

Rag

痛点

(1)知识的局限性：模型自身的知识完全源于它的训练数据，而现有的主流大模型（ChatGPT、文心一言、通义千问…）的训练集基本都是构建于网络公开的数据，对于一些实时性的、非公开的或离线的数据是无法获取到的，这部分知识也就无从具备。

(2)幻觉问题：所有的AI模型的底层原理都是基于数学概率，其模型输出实质上是一系列数值运算，大模型也不例外，所以它有时候会一本正经地胡说八道，尤其是在大模型自身不具备某一方面的知识或不擅长的场景。而这种幻觉问题的区分是比较困难的，因为它要求使用者自身具备相应领域的知识。

(3)数据安全性：对于企业来说，数据安全至关重要，没有企业愿意承担数据泄露的风险，将自身的私域数据上传第三方平台进行训练。这也导致完全依赖通用大模型自身能力的应用方案不得不在数据安全和效果方面进行取舍。

rag技术总结：

RAG（中文为检索增强生成） = 检索技术 + LLM 提示。

rag实现过程

完整的RAG应用流程主要包含两个阶段：

数据准备阶段：数据提取——>文本分割——>向量化（embedding）——>数据入库

应用阶段：用户提问——>数据检索（召回）——>注入Prompt——>LLM生成答案

优点：

提供多样化的上下文：因为你可以从不同的角度查看用户的查询，所以顶级结果里会出现更多样化的内容片段。与专注于单一视角的内容段落相比，你更有可能看到能够涵盖话题多个方面的内容作为顶级结果出现。

额外的控制层面：像其他机器学习解决方案一样，RAG融合提供了额外的控制手柄，让你可以微调你的应用，并让它更加符合你的期望目标。

自动校正：通过使用LLM作为用户在文本框中输入内容与实际在数据库中搜索内容之间的中间人，你可以纠正拼写错误，添加与用户查询相关的上下文信息（关于用户的信息、时间、他们的账户状态等），以及/或潜在地筛选特定类型的内容。

成本：这是一个有些争议的问题，因为成本既是RAG融合的优点也是缺点，让我来解释一下。你大概知道，向量搜索比LLMs要便宜得多，那么用于RAG融合的额外LLM调用是不是应该会增加应用的整体成本呢？不过……让我们再来看看LLM的成本。你大概也知道，你可能使用的大多数主流LLMs都采用基于令牌的计费模式。即使是自己托管LLM，你的成本也会与处理的令牌数量大致成正比。我们基本上在这儿向LLM发送两次请求，一次大概有100个令牌用来生成相似的查询，另一次则会有数千个令牌，我们提供相关文本块并希望从LLM那里得到适当的回应。所以基本上我们第一次对LLM的调用要比第二次便宜10倍到100倍。所以即使RAG融合每10次查询节省一次后续问题，你在成本上还是能领先的。

缺点：

延迟:正如你现在可能知道的，LLMs需要大量的计算资源，因此它们的速度相对较慢(相对于我们应用程序中的其他部分)。根据你的应用程序，向LLM发送一次额外的请求可能会让用户的体验变得糟糕，因为他们可能需要等待几百毫秒的时间。

自动纠错:是的，这是一个优点，但是当它不起作用时，也可能是一个缺点。这通常发生在你的内容包含内部术语或行话，而这些术语或行话没有出现在LLM学习过的数据中。在这种情况下，LLM可能会出现困惑并生成完全无关的查询，从而影响到最后的结果。

成本:正如我们之前讨论的，如果RAG融合对你应用程序的整体效益贡献不大，你最终可能会花费更多的费用，但收益却很有限。

rag更多具体实现细节参考:

[一文读懂：大模型RAG（检索增强生成）含高级方法](https://www.zhihu.com/tardis/zm/art/675509396?source_id=1003)

蒸馏：

第三方开源蒸馏deepseek r1项目

[open-thoughts/open-thoughts：Open Thoughts：思维模型的完全开放数据管理](https://github.com/open-thoughts/open-thoughts?tab=readme-ov-file)

蒸馏过程讲解

[使用DeepSeek-R1蒸馏训练自己的本地小模型（Qwen2.5-0.5B），原理流程全讲解，模型数据全给你](https://mp.weixin.qq.com/s/P12FAEFHQ_6aRPHlXKGd-w?token=431651514&lang=zh_CN)

理论部分：

DeepSeek-R1的蒸馏原理非常简单，不像传统的知识蒸馏那样通过教师模型和学生模型的KL散度或交叉熵损失来调整学生模型，其采用了一种更加简单高效的蒸馏方法，即通过DeepSeek-R1产生的数据去微调（SFT）其他模型（学生模型），注入DeepSeek-R1（教师模型）的相关知识。并且DeepSeek声称这种方法得到的小模型效果优于小模型直接进行强化学习。

Step1

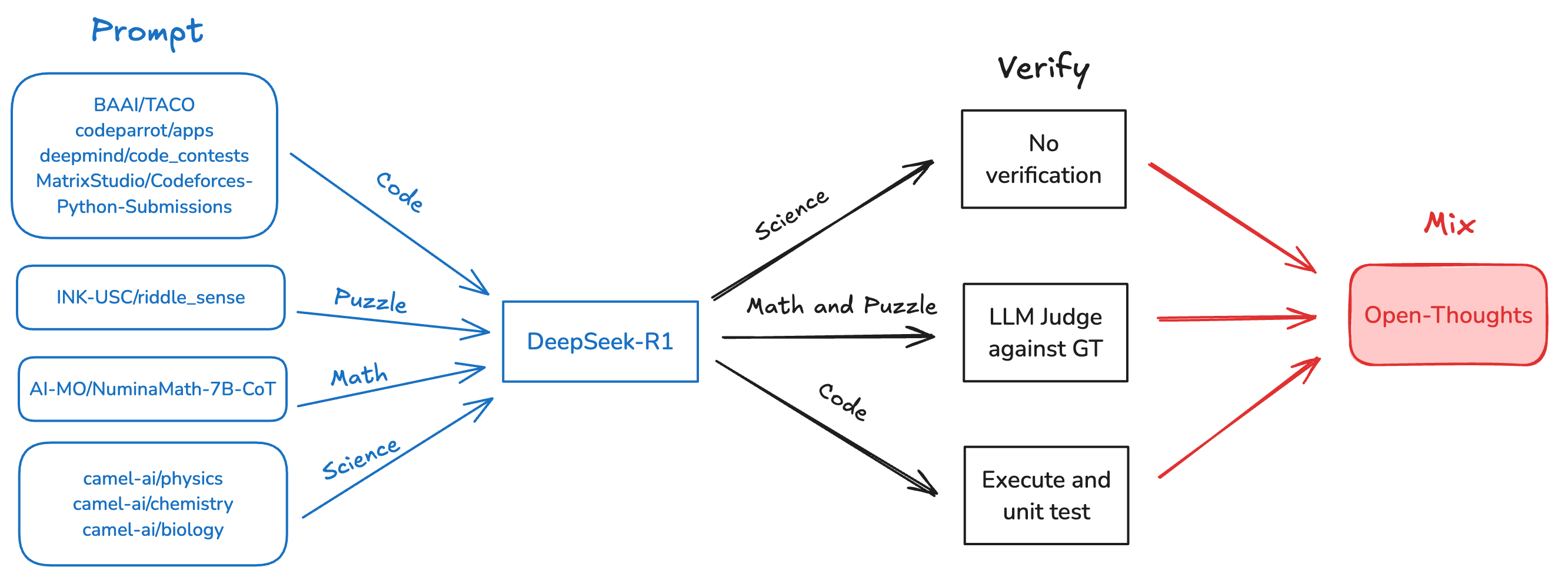
将不同领域的数据（代码、数学、科学、谜题）输入到DeepSeek-R1模型中，生成带思考过程的答案。

Step2

对生成的结果进行筛选（过滤掉一些不正确的答案），对于科学类的问题（开放性问答），无需验证，对于数学和谜题类的问题，通过大模型对生成结果和标准答案进行评判，对于代码类的问题，通过执行代码通过与否进行判断。

Step3

将所有数据进行混合，得到最终的数据集。



微调：

监督微调（SFT）概述：

SFT 是一种改善和定制预训练 LLM 的方法，通过在较小的数据集上重新训练基础模型，使其能够更好地执行指令和回答问题。

SFT 可以提高模型的整体性能，增加新知识，或适应特定任务和领域。

SFT 技术：

全微调：对预训练模型的所有参数进行重新训练，效果最佳但需要大量计算资源。

LoRA（低秩适应）：只训练小的适配器，减少内存使用和训练时间，非破坏性。

QLoRA：LoRA 的扩展，提供更高的内存节省，适合 GPU 内存受限的情况，但训练时间更长。

[FAQ + Is Fine-tuning Right For Me? | Unsloth Documentation](https://docs.unsloth.ai/get-started/beginner-start-here/faq-+-is-fine-tuning-right-for-me" \l "does-fine-tuning-add-new-knowledge-to-my-model)

微调的好处

微调可以做 RAG 能做的所有事情，但 RAG 不能 Fine-tuning 可以通过在训练期间将外部知识直接嵌入到模型中来复制 RAG 的功能，使其能够在不依赖外部系统的情况下执行诸如回答小众问题或总结文档等任务。微调还可以将上下文和模式集成到模型中，模拟检索行为。

特定任务的掌握微调将有关域或任务的深入知识直接嵌入到模型中，使其能够高精度地处理结构化、重复或细微差别的查询，这是 RAG 无法单独实现的。

独立于检索微调模型无需外部数据即可有效运行，即使在检索系统出现故障或知识库不完整时也能确保无缝性能。

更快的推理微调模型无需检索步骤即可提供直接响应，使其成为速度至关重要的场景的理想选择。

自定义行为和语气微调允许精确控制模型的行为和通信方式，确保与品牌声音、法规要求或其他约束保持一致。

回退稳健性在组合系统中，微调模型可确保可靠任务性能的基线水平，即使 RAG 系统检索不相关或不完整的信息也是如此。

Fine-tuning 是否为我的模型添加了新知识？

是的，一点没错！一个常见的误解是微调不会将新知识引入模型，但事实并非如此。事实上，微调的真正目的是向模型传授全新的概念或知识，只要您的数据集包含相关信息。微调使模型能够在获得特定数据时从头开始学习。

RAG 的性能比 Fine-tuning 好吗？

另一个普遍存在的误解是，RAG 在基准测试中的表现始终优于微调。这是不正确的。如果微调做得好，与 RAG 相比，它通常会获得更好的结果。相反的陈述通常源于不正确的实现 - 例如错误配置 LoRA 参数或普遍缺乏微调经验。

Unsloth 通过自动为您选择最佳参数配置来处理这些复杂性。您所需要的只是一个高质量的数据集，您将获得一个经过微调的模型，该模型可以发挥其最佳性能。

结合 RAG + 微调

这就是为什么我们建议用户将 RAG 和 fine-tuning 组合在一起，而不是使用其中一种。RAG 通过动态访问外部知识来增强适应性，而微调则加强了系统的核心专业知识，确保其可靠执行而不会过度依赖检索。此外，微调使模型能够更有效地解释和整合检索到的信息，从而产生无缝且上下文准确的响应。

为什么你应该结合RAG和微调呢？

特定于任务的专业知识：微调擅长于特定任务，而 RAG 则动态检索最新或外部知识。它们共同处理核心和特定于上下文的需求。

适应性：微调模型在检索失败时提供稳健性，RAG 使系统无需不断重新训练即可保持最新状态。

效率：微调建立了基线，而 RAG 通过处理动态细节减少了对详尽训练的需求。

微调节省显存方式

[Unsloth微调Llama3-8B，提速44.35%，节省42.58%显存，最少仅需7.75GB显存 - 大模型知识库|大模型训练|开箱即用的企业大模型应用平台|智能体开发|53AI](https://www.53ai.com/news/qianyanjishu/1755.html)

但unsloth暂不支持cpu微调

[我可以使用 Unsloth 在 CPU 上微调模型吗？ ·问题 #1106 ·unslothai/unsloth](https://github.com/unslothai/unsloth/issues/1106)

[(99+ 封私信 / 80 条消息) 深度学习训练模型时，GPU显存不够怎么办？ - 知乎](https://www.zhihu.com/question/461811359)

大模型微调技术：

[huggingface/peft： 🤗 PEFT： 最先进的参数高效微调。](https://github.com/huggingface/peft)

R1 推理模型训练过程：

[在本地训练您自己的 R1 推理模型 （GRPO）](https://unsloth.ai/blog/r1-reasoning)

[Unsloth训练自己的R1推理模型 - DeepSeek GRPO](https://www.youtube.com/watch?v=aNisiIo1xBY)

将 Llama 3.1 （8B） 转换为推理模型：

[https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.1\_（8B）-GRPO.ipynb](https://colab.research.google.com/github/unslothai/notebooks/blob/main/nb/Llama3.1_(8B)-GRPO.ipynb)

量化：

痛点：

近年来，随着Transformer、MOE架构的提出，使得深度学习模型轻松突破上万亿规模参数，从而导致模型变得越来越大，因此，我们需要一些大模型压缩技术来降低模型部署的成本，并提升模型的推理性能。 模型压缩主要分为如下几类：

剪枝（Pruning）

知识蒸馏（Knowledge Distillation）

量化Quantization）

模型量化是指以较低的推理精度损失将连续取值（通常为float32或者大量可能的离散值）的浮点型权重近似为有限多个离散值（通常为int8）的过程。通过以更少的位数表示浮点数据，模型量化可以减少模型尺寸，进而减少在推理时的内存消耗，并且在一些低精度运算较快的处理器上可以增加推理速度。

[大模型量化概述 - 知乎](https://zhuanlan.zhihu.com/p/662881352)

deepseek未量化前671b，文件体积高达 720GB

deepseek完整版1.73-bit动态量化后模型 169GB

[Huzderu/deepseek-r1-671b-1.73bit](https://ollama.com/Huzderu/deepseek-r1-671b-1.73bit)

硬件需求

部署此类大模型的主要瓶颈是内存+显存容量，建议配置如下：

DeepSeek-R1-UD-IQ1\_M：内存 + 显存 ≥ 200 GB

除了模型参数占用的内存+显存空间（158 GB ）以外，实际运行时还需额外预留一些内存（显存）空间用于上下文缓存。预留的空间越大，支持的上下文窗口也越大。

[完整的671B MoE DeepSeek R1怎么塞进本地化部署?详尽教程大放送|内存|gpu|上下文\_网易订阅](https://www.163.com/dy/article/JNDEHFII0511AQHO.html)

部署推理模型：

大模型部署推理框架对比

ollama部署推理支持cpu+gpu推理方式，它将尝试将尽可能多的 LLM 层卸载到 GPU 上，然后如果它不适合，其他层将在 CPU 上运行。

[Ollama 可以同时使用 CPU 和 GPU 进行推理吗？ ·问题 #3509 ·OLLAMA/OLLAMA](https://github.com/ollama/ollama/issues/3509)

deepseek运行温度参数参考

来源：[unsloth/DeepSeek-R1-Distill-Qwen-32B-bnb-4bit ·拥抱脸](https://huggingface.co/unsloth/DeepSeek-R1-Distill-Qwen-32B-bnb-4bit)

注意：我们建议在运行这些模型时设置适当的温度（介于 0.5 和 0.7 之间），否则您可能会遇到无休止重复或输出不连贯的问题。