llama3技术报告-后训练

后训练

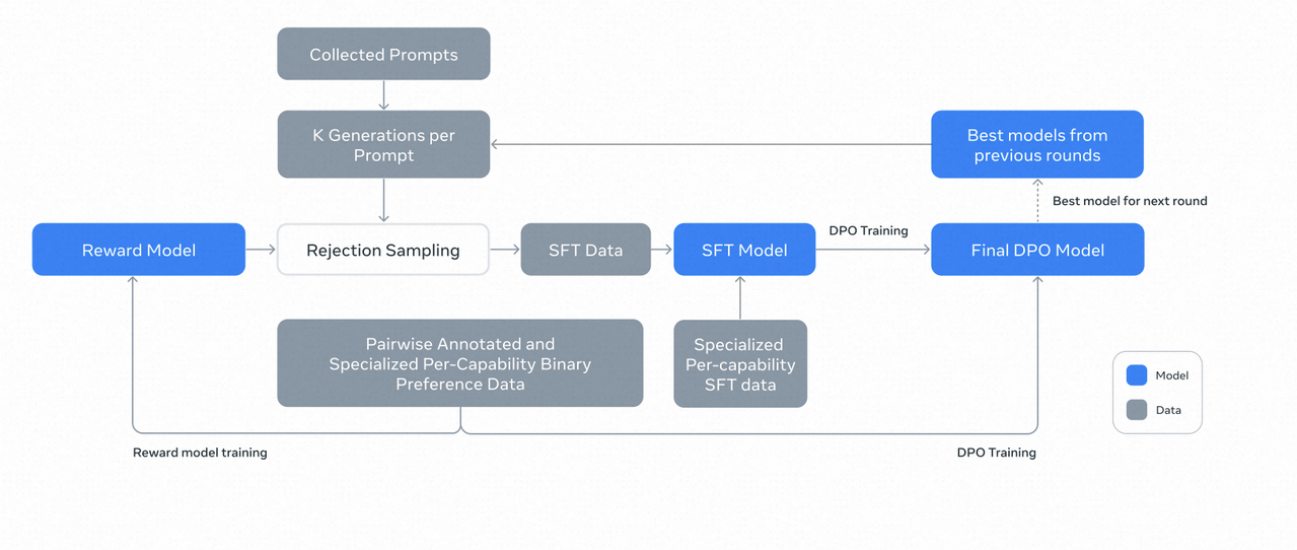

Figure 7 Illustration of the overall post-training approach for Llama 3. Our post-training strategy involves rejection sampling, supervised finetuning, and direct preference optimization. See text for details.

每轮后训练需要在通过人工注释或合成生成的示例上进行SFT和DPO。首先基于标注好的pairwise数据在预训练模型的基础上训练奖励模型，并筛选sft数据。然后用SFT在预训练模型进行微调，基于pairwise数据通过DPO与预训练模型对齐。

方法建模

1. **聊天对话格式**：为了训练LLM与人类的交互，为模型定义一个聊天对话协议，以理解人类指令并执行对话任务。此外Llama3支持工具的使用 (在一个对话回合中，可能生成多个消息并被发送到不同的位置（如用户、ipython)，定义了一个聊天对话协议，使用各种特殊的头标记和终止标记。头标记用于指示对话中每条信息的来源和目的地。终止标记用于指示何时人类和AI交替发言的时间。（本质就是定义了新的template，针对tool using 的调用设计了特殊的token）
2. **奖励模型**：在预训练模型的基础上训练一个奖励模型，训练目标与llama2相同，只是去掉了损失中的边际项，因为观察到data scaling后的改进效果递减。首先过滤掉偏好数据中响应相似的样本，每个偏好数据都包含2-3个回复（edited>chosen>rejected）。训练过程中，将prompt和多个回复拼在一起，回复的顺序随机。
3. **SFT**：使用奖励模型对人类注释的prompt进行拒绝采样，连同这些拒绝采样的数据和其他数据源（包括合成数据），使用标准交叉熵损失（同时屏蔽prompt上的损失）对预训练语言模型进行SFT。在405B模型 8.5K 到 9K 步的过程中以 1e-5 的学习率进行微调。

4. **DPO**：用DPO对SFT模型进行训练，与人类偏好对齐(DPO本质是二分类问题，调整模型参数鼓励模型输出Good Response，不输出Bad Response)，使用之前对齐中表现最好的模型收集的最新一批偏好数据。相比于PPO，DPO对计算量更低，且表现更好，学习率1e-5，将β超参数设置为0.1。此外去除头标记和尾标记，这些token与学习目标冲突（因为好回复和坏回复中都有）；对好回复应用额外的负对数似然 (NLL) 损失项，其缩放系数为 0.2，保持所需的生成格式并防止所选响应的对数概率降低。
5. **模型平均**：对从不同数据和超参数训练的RM，SFT和DPO模型进行聚合。

- Model merging：直接做模型加权平均，这会导致某些模型的知识被抹除掉。Tie则是首先确定每个参数的偏移方向（也就是最大改变的方向），然后在该方向做单位偏移。Spherical：只考虑每个方向的角度，不考虑偏移长度。

后训练数据

pairwise偏好数据

在每轮之后对部署多个模型进行注释，针对每个用户prompt利用两个模型生成两个回复。这些模型是基于不同数据混合和对齐方法训练的，以保障数据的多样性。偏好排名过程中，标注者以四个评级表示其对prefer相较于rejected的评价：明显更好、更好、稍微更好或稍微更好。此外还在排名后，允许注释者编辑回答，因此部分偏好数据中，有一部分有三个响应排序（编辑 > 选择 > 拒绝）。

与llama2相比，prompt和回复的平均长度增加，说明能处理更复杂的任务。对于奖励建模和DPO，只采用明显更好和更好的偏好数据并删除掉回复相似的数据，奖励模型训练使用所有的偏好数据，DPO只用最新的batch。

基于pairwise偏好数据可以训练出奖励模型。

Dataset	% of comparisons	Avg. # turns per dialog	Avg. # tokens per example	Avg. # tokens in prompt	Avg. # tokens in response
General English	81.99%	4.1	1,000.4	36.4	271.2
Coding	6.93%	3.2	1,621.0	113.8	462.9
Multilingual	5.19%	1.8	1,299.4	77.1	420.9
Reasoning and tools	5.89%	1.6	707.7	46.6	129.9
Total	100%	3.8	1,041.6	44.5	284.0

SFT数据

拒绝采样：对每个用户的prompt，从最新的聊天模型策略中采样K（通常在10到30之间）个输出（通常是最好的性能检查点来自上一次后训练迭代，或特定能力的最好性能检查点），利用奖励模型选择最好的一个。在后训练的后期轮次中，我们引入系统提示，以引导拒绝采样响应符合不同能力所需的期望语调、风格或格式。 每个 SFT 示例由一个上下文(即除最后一轮外的所有对话轮次）和一个最终response组成。SFT和偏好数据有重复的领域，但是按照不同方法产生的数据。

数据质量控制

- 数据清洗：实施了一系列基于规则的数据移除和修改策略，例如，为了缓解过于道歉的语气问题，我们识别出过度使用的短语（如“I’ m sorry”或“I apologize”），并仔细平衡我们数据集中这类样本的比例。
- 主题分类：微调8B的模型为主题分类器，进行粗粒度分类（“数学推理”）和细粒度分类（“几何和三角学”）
- 质量评分：分别采用RM和llama信号进行质量评分，这两个评分不一致性很高，选取二者的并集。
- 难度评分：分别采用Instag和llama评分，选取最复杂的数据。
- 语意去重：使用RoBERTa对完整对话进行聚类，每个聚类内按质量分数×难度分数排序，采用贪心的思想，只保留与迄今为止聚类中看到的示例的最大余弦相似度小于阈值的示例。

特殊能力

代码能力

在预训练模型上，使用1Ttoken的代码数据上训练一个代码专家。后续的做代码数据生成在代码专家上做。

1. 执行反馈：观察到小模型在更大模型生成的数据上进行训练会带来性能提升，但是405B在自己生成的数据上训练并没有帮助。为解决这个限制，我们引入执行反馈作为真实性的来源，使模型能够从错误中学习并保持在正确的轨道上，采用如下流程生成了大约一百万合成编码对话的大型数据集：
- 问题生成：生成了一个涵盖广泛话题的编程问题描述的集合，
- 代码生成：使用llama3生成代码，观察到，在提示中添加通用的良好编程规则可以提高生成的解决方案质量。此外，我们发现要求模型在注释中解释其思维过程是有帮助的。
- 正确性分析：将所有生成的代码通过解析器和linter运行，确保语法正确性；对于每个问题和解决方案，我们提示模型生成单元测试，在容器化环境中与解决方案一起执行，捕捉运行时执行错误和一些语义错误。
- 错误反馈与自我修正：当代码出错时，提示模型进行修正，提示包括原始问题描述、错误的解决方案以及语法错误和执行错误将没有问题的对话加入到SFT的数据集中。

2. 编译语言翻译：将常见编程语言的数据翻译成较少使用的语言来补充现有数据，以提升在较少使用编程语言上的性能。
3. 回译：首先收集大量正确的代码，然后：
- 要求llama3根据给定的代码生成注释或者解释一段代码
- 根据生成的文字要求模型写出代码
- 比较原始代码和生成代码之间的距离，越接近越好。

多语言

在预训练模型的基础上，在包含90%多语言标记的数据上进行预训练，来训练一个多语言专家。然后再进行后训练。总体分布是2.4%人类注释，44.2%来自其他NLP任务的数据，18.8%拒绝抽样数据，

和34.6%翻译推理数据。

数学和推理

论文将推理定义为执行多步骤计算并得出正确最终答案的能力。面临的难题：

1. 缺少prompt和query。2. 缺少真实的COT。3. 中间步骤不正确。4. 教会模型使用外部工具。5. 训练与推理差异。

采用以下手段：

1. 从已有的数学数据中构建问答对，用于SFT。此外对于模型表现不佳的数学技能，让人类来来构建更多的prompt和query来教授模型。
2. 用Llama 3为一组prompt生成逐步解决方案，对于每个prompt生成很多方案，再用llama3进行每个方案的正确性。训练更细粒度的RM，为每一步骤都评价其正确性。
3. 通过文本推理和相关Python代码的组合来解决推理问题。代码执行被用作反馈信号，以消除推理链无效的情况，确保推理过程的正确性。
4. 为了模拟人类反馈，我们利用错误的生成，并通过提示Llama 3产生正确的生成来进行错误纠正。使用错误尝试的反馈并纠正它们的迭代过程有助于提高模型准确推理并从错误中学习的能力。

长文本

利用合成数据增强长文本能力。

- 问答对：从预训练数据中提取一组长文档，拆分成8k token的块，使用过往版本的llama 基于随机选择的块生成QA对，在训练期间，整个文档被用作上下文。
- 摘要：对于长文本采用分层摘要，首先使用Llama 3 8K上下文模型对8K输入长度的块进行摘要，然后对摘要进行二次摘要。在训练期间提供完整文档，并prompt模型在保留所有重要细节的同时摘要文档。我们还基于文档摘要生成QA对，并用需要对整个长文档有全局理解的问题prompt模型。
- 观察到代码往往也都是很长的文本，所以通过解析Python文件以识别import语句并确定它们的依赖关系，提升模型的长文本能力。选择最经常被import的文件，并随机删除一些文件，然后prompt模型识别依赖于缺失文件的文件，并生成必要的缺失代码。

观察到使用0.1%的合成长文本数据，就能在短文本环境和长文本环境基准测试中优化性能。此外观察到只要SFT模型的训练过程中使用了长文本数据，DPO就不需要长文本数据。

工具使用

教会LLM能使用搜索引擎，解释器等工具，使得LLM不再只是一个对话模型。可以在工具调用的模板中加入一些特殊的token，在prompt中加入这些token，从而调用这些Funtion calling。

此外还增强了zero-shot工具使用能力，即在上下文中未见过的工具和用户查询，训练模型生成正确的工具调用。

实现：核心的工具由具有不同方法的Python对象实现，zero-shot工具可以作为具有描述、文档（即如何使用它们的示例）的Python函数实现，模型只需要函数的签名和文档字符串作为上下文来生成适当的调用。我们还将函数定义和调用转换为JSON格式，例如，用于Web API调用。所有工具调用都由

Python解释器执行，该解释器必须在Llama 3系统prompt中启用。核心工具可以在系统prompt中单独启用或禁用。

数据采样：

- 单步工具使用：我们首先通过少量样本生成合成用户prompt，这些prompt在构建时就需要调用我们的核心工具之一（例如，超出我们知识截止日期的问题）。然后，仍然依赖少量样本生成，我们为这些prompt生成适当的工具调用，执行它们，并将输出添加到模型的上下文中。最后，我们再次提示模型根据工具的输出生成对用户查询的最终答案。我们最终得到以下形式的轨迹：系统prompt、用户prompt、工具调用、工具输出、最终答案。
- 多步工具使用：遵循类似的协议，首先生成合成数据以教授模型基本的多步工具使用能力。为此，我们首先提示Llama 3生成至少需要两次工具调用的用户prompt，这些可以是来自我们核心集合的相同或不同的工具。然后，基于这些prompt，few-shot提示Llama 3生成一个解决方案，该方案由交错的推理步骤和工具调用组成。

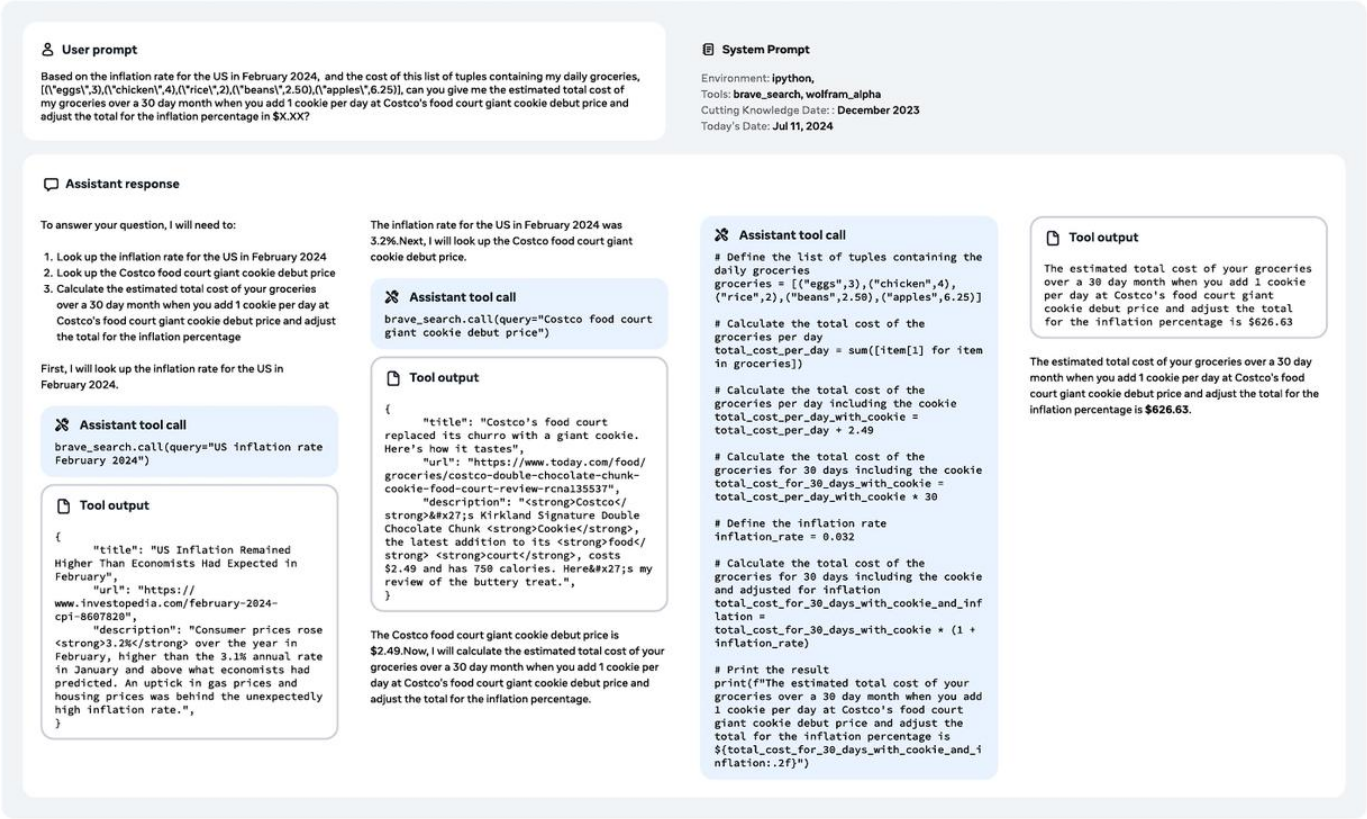

Figure 10 Multi-step tool usage. Example of Llama 3 performing multi-step planning, reasoning, and tool calling to solve a task.

- 文件：对csv,pdf,txt等格式的文件进行注释，我们的prompt基于提供的文件，并要求概述文件内容、查找和修复错误、优化代码片段、执行数据分析或可视化。

模型幻觉

幻觉：模型往往过于自信，即使在它们知之甚少的领域也是如此。

构建没有幻觉的数据集，对于关于模型训练之后发生的事情的问题，教会模型不要回答。后训练应该使模型与“知道自己知道什么”对齐，采用一种知识探测的方式：

- 从预训练数据中提取数据片段。
- 通过提示Llama 3生成关于这些片段（上下文）的事实性问题。
- 从Llama 3获取问题的回答样本。
- 使用原始上下文作为参考，以Llama 3为裁判（这里感觉还涉及人，因为对于训练完成后的事情llama 3也不知道对不对），对生成的正确性进行评分。
- 使用Llama 3为裁判，对生成的信息性进行评分。
- 对于在生成中信息丰富但不正确的回答，使用Llama 3生成拒绝回答。

此外，预训练数据并不总是事实一致或正确的。因此，我们还收集了有限的标记事实性数据，这些数据涉及存在事实矛盾或错误陈述的敏感话题。

可引导性

可引导性：模型的行为和结果引导至满足用户instruct的能力。通过系统提示和自然语言指令增强其可引导性，特别是在回答长度、格式、语气和角色/个性方面。

构建数据时，需要设计各种系统提示，标注者与模型进行对话，评估它们在对话过程中是否遵循系统提示。基于构建好的数据，再进行奖励建模，SFT，DPO训练。