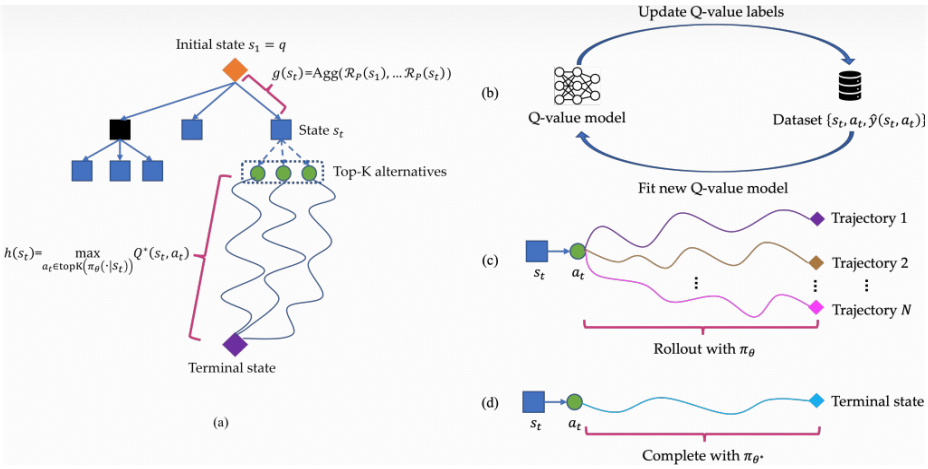
知乎



2024南洋理工：探索openai神秘的Q*项目，百倍提升7B模型的推理能力

SmartMindAI
专注搜索、广告、推荐、大模型和人工智能最新技术，欢迎关注我
已关注

Introduction

大型语言模型（LLMs）在自然语言推理任务中表现出色，包括数学问题、代码生成和计划。然而，LLMs在推理步骤增多时容易引入错误和不一致的陈述。目前的方法主要集中在提升LLMs的"系统1"能力，即快速但不准确的思考模式。解决复杂推理问题需要更深入、深思熟虑和逻辑性强的"系统2"思考方式。现有的深思方法需要针对每个任务进行人工设计，且在处理多步推理问题时速度较慢。为了解决这些问题，我们提出了Q，一种通用的、多样化的和敏捷的框架，通过规划深入指导提高LLMs的多步推理能力。与现有方法不同，Q无需依赖专有知识来设计启发函数，并且可以在无需微调LLMs的情况下解决各种任务。我们的实验证明，Q显著提高了现有LLMs的多步推理能力。

Preliminary

Formulate the Multi-step Reasoning of LLMs as an MDP

以输入q为例，LLM的回答生成流程可以分解为多个推理步骤。在最终答案序列$\mathbf{a}$中，这些单步骤的推理可以被视为串联起来。 $\mathbf{a} = [a_1; a_2; \ldots; a_T]$

每个步骤可以是一行输出或是由LLM输出的固定数量的标记。我们将LLM的多步骤推理流程视作马尔可夫决策过程（MDP）。 $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ 其中状态s_t表示已生成的推理痕迹与输入问题的串联。动作a_t表示LLM生成的下一轮推理步骤。状态迁移由连接操作完成。奖励函数R度量问题解决程度，折现因子为γ。奖励函数通过比较最终结果与真实值来给出奖励。

$$\mathcal{R}(s_t, a_t) = \begin{cases} 1 & t = T \wedge [s_t; a_t] \text{ matches the ground-truth} \\ 0 & \text{otherwise} \end{cases}$$

尤其是，在代码生成任务中，如果生成的代码通过所有测试用例，则奖励为1。在数学推理任务中，如果最终答案与真实答案相符，则奖励也为1，这是之前研究中常用的方法。 最后，LLM生成的策略π_θ会产生基于输入问题的推理序列：

$$\pi_\theta(a_t|s_t) = \text{LLM}(a_t|s_t),\ \pi_\theta(\mathbf{a}|q) = \prod_{t=1}^{T} \pi_\theta(a_t|s_t).$$

给定MDP和策略π_θ，状态-动作对(s_t, a_t)的价值由Q函数表示。

$$Q^{\pi_\theta}(s_t, a_t) = \mathbb{E}_{\pi_\theta}\left[\sum_{t'=t}^{T} \gamma^{T-t'} \mathcal{R}(s_{t'}, a_{t'})\right]$$

知乎

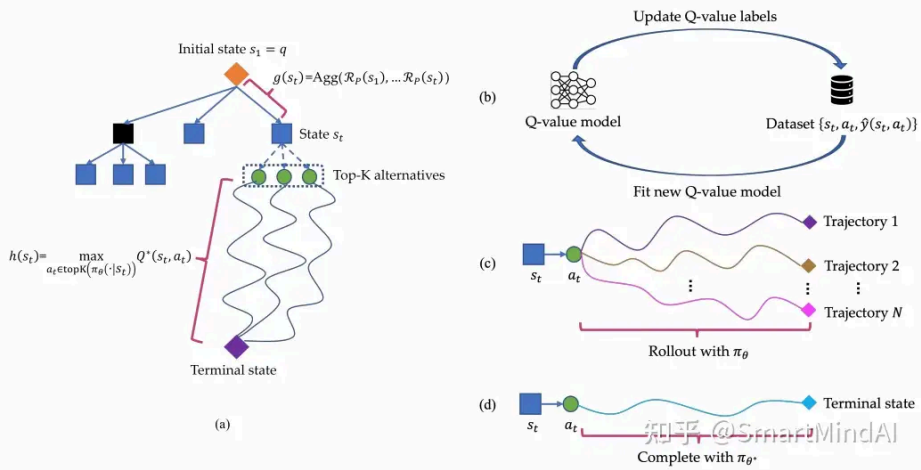
$$Q^*(s_t, a_t) = \mathcal{R}(s_t, a_t) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q^*(s_{t+1}, a_{t+1})$$

这表示从状态s_t开始，采取动作a_t后，按照最优策略π^*的价值。

A Search

A是一种重要的启发式搜索算法，最初用于路径规划问题。它通过计算每个顶点的f值来寻找最短路径。f值由累计路径成本g(n)和启发式值h(n)组成。A采用最优首先搜索策略，选择f值最小的顶点进行探索，直到达到目标点。当启发式函数h(·)是可接受的时，A可以找到最优路径。

Q: A General, Versatile and Agile Deliberation Framework for LLMs



现代预训练语言模型以自动回归方式生成自然语言，即根据之前生成的令牌预测下一个令牌。然而，在多步推理问题中，如果之前的步骤有错误，模型可能会引入错误和不一致的陈述，导致问题无法解决。为了解决这个问题，我们引入了Q，一个基于A的决策框架，可以指导模型选择最佳的下一步。我们将问题的推理序列视为一个启发式搜索过程，每个状态都有一个f值来估计扩展该状态的收益。

$$f(s_t) = g(s_t) + \lambda h(s_t),$$

其中$g(s_t)$表示从初始状态s_1累积的收益，$h(s_t)$是用于衡量从状态s_t到正确答案的概率的启发式值，λ是用于平衡$g(s_t)$和$h(s_t)$项重要性的系数。具体来说，我们使用基于过程的奖励函数$\mathcal{R}_P$来计算累积的收益$g(s_t)$，该函数编码推理任务的先验知识或偏好。

$$g(s_t) = \mathbf{Agg}(\mathcal{R}_P(s_1), \ldots, \mathcal{R}_P(s_i), \ldots, \mathcal{R}_P(s_t)),$$

其中 Agg $\in \{\min, \max, \sum, [-1]\}$

使得$[-1]$表示将路径从s_1到s_t的最后一个状态的奖励作为收益；以及s_{i-1}是s_i的前缀，对于$1 < i \leq t$。基于过程的奖励函数$\mathcal{R}_P$可以从人类反馈、真值、规则或者推理步骤的logits中学习，这些logits反映了LLM的置信度。同时，我们使用状态s_t的最优Q值（参见等式 ()）作为启发式值$h(s_t)$。换句话说，f-值定义为：

$$f(s_t) = g(s_t) + \lambda \max_{a_t \in \mathcal{A}} Q^*(s_t, a_t).$$

由于无法列举所有可能的下一步推理步骤，实际上我们只考虑LLM返回的前K个候选步骤。因此，等式可以简化为

$$f(s_t) = g(s_t) + \lambda \max_{a_t \in \text{top-K}(\pi_\theta(\cdot|s_t))} Q^*(s_t, a_t)$$

Estimation of Optimal Q-value

知乎

$D = \{q_i, \{\mathbf{a}_{i_j}\}_{j=1}^{M}\}_{i=1}^{N}$

通过学习代理Q值模型$\hat{Q}$来近似真实的Q^*，其中q_i是训练问题

$\mathbf{a}_{i_j} \sim \pi_\theta(\cdot|q_i)$

这是从LLM策略中采样的轨迹。具体定义如下：

$\hat{Q} = \arg\min_Q \frac{1}{NMT} \sum_{i=1}^{N} \sum_{j=1}^{M} \sum_{a_t \in \mathbf{a}_{i_j}} \left(Q(s_t, a_t) - \hat{y}(s_t, a_t)\right)^2$

其中 $s_t = [q_i; a_1; \ldots; a_{t-1}]$

$\mathbf{a}_{i_j}$路径中的截至$t-1$时间步骤的推理部分轨迹，$\hat{y}(s_t, a_t)$是近似真实最优Q值的标签，即$Q^*(s_t, a_t)$。

离线强化学习。

我们使用离线数据集D来学习代理Q值模型$\hat{Q}$。每一次迭代ℓ，我们构建Q值标签。

$$\hat{y}_\ell(s_t, a_t) = \begin{cases} \mathcal{R}(s_t, a_t) & t = T \\ \mathcal{R}(s_t, a_t) + \gamma \max_{a_{t+1} \in \text{top-K}(\pi_\theta(\cdot|s_{t+1}))} \hat{Q}_{\ell-1}(s_{t+1}, a_{t+1}) & \text{otherwise} \end{cases}$$

我们使用$\hat{Q}_{\ell-1}$作为代理Q值模型，并根据等式训练新的代理模型$\hat{Q}_\ell$。我们交替进行L次迭代，并在执行决策时使用$\hat{Q}_L$作为代理Q值模型。

从展开学习。

从状态动作对(s_t, a_t)出发，我们通过策略π_θ进行多次展开或MCTS，得到轨迹池P。然后，我们选择累积奖励最高的最佳推理序列作为Q值标签。

$\hat{y}(s_t, a_t) = \mathcal{R}(s_t, a_t) + \max_{\substack{\tau \sim P \\ (s_{t'}, a_{t'}) \in \tau}} \left[\sum_{t'=t+1}^{T} \gamma^{T-t'} \mathcal{R}(s_{t'}, a_{t'})\right]$

利用更强的LLM逼近最优政策。

利用最优Q值来源于最优策略这一事实，如果可用（例如GPT-4），可以通过使用另一个更强的LLM来完成状态动作对的轨迹，以此来估计该状态动作对的最优Q值：π_{θ^*}

$\hat{y}(s_t, a_t) = \mathcal{R}(s_t, a_t) + \sum_{t'=t+1}^{T} \gamma^{T-t'} \mathcal{R}(s^*_{t'}, a^*_{t'})$

其中 $s^*_{t'} = [s_t; a_t; a^*_{t+1}; \ldots; a^*_{t'-1}]$

以及 $a^*_{t'} \sim \pi_{\theta^*}(\cdot|s^*_{t'})$

Deliberative Planning with A

得到代理Q值模型$\hat{Q}$后，我们可以计算每个状态的f值，并使用A进行最优优先搜索。算法概述了决策规划过程。

知乎

Input: question q, LLM policy π_θ, proxy Q-value model $\hat{Q}$
1: $unvisited \leftarrow \{q\}, visited \leftarrow \emptyset$
2: **while** $unvisited \neq \emptyset$ **do**
3: $s \leftarrow \arg\max_{s' \in unvisited} f(s')$
4: $unvisited \leftarrow unvisited \backslash \{s\}, visited \leftarrow visited \cup \{s\}$
5: **if** s is a terminal state **then return** the complete trajectory s
6: **for each** $a \in$ top-K$(\pi_\theta(\cdot|s))$ **do**
7: $s' \leftarrow [s; a]$
8: **if** $s' \notin visited$ **then** $unvisited \leftarrow unvisited \cup \{s'\}$

Experimental Settings

实现细节。 Q方法的实现包括三个步骤：1) Q值估算；2) 实用性聚合；3) 轨迹选择。对于Q值估算，我们使用随机滚动播放或蒙特卡洛树搜索*来收集一组轨迹，并选择累计奖励最高的路径作为Q值标签。实用性聚合使用不同的函数来计算每个推理步骤的中间信号。对于轨迹选择，我们根据最高累计奖励来选择最佳推理路径。在实验中，我们设置了一些参数，如温度和动作数量。最后，我们根据是否生成正确答案或通过测试用例来赋予奖励。

Experimental Results

Table 1: Statistics of datasets.

Dataset	GSM8K	MATH	MBPP
Domain	Math Reasoning	Math Reasoning	Code Generation
Training	5000	8000	374
Testing	1319	5000	500
Average Steps	4.5	11.0	7.0

GSM8K。

在GSM8K数据集的对比中，我们选择了Llama-2-7b作为基础模型，经过在MetaMath上微调后，准确率达到65.2%。然后，我们将微调后的Llama-2-7b作为策略π_θ，使用随机滚动播放来收集Q值模型（QVM）的训练标签。 对于效益聚合，我们在PRM800K上训练了过程奖励模型（PRM），为每个推理步骤提供中间信号。 使用PRM和QVM进行验证时，我们发现无论是在验证还是对齐方面，Q方法都表现更好。 在基于规划的方法比较中，我们发现使用常量聚合效益的Q方法仍然能够超过N选一方法，即使在相同系列的大规模语言模型（LLM）中也是如此。 通过在PRM800K上训练的PRM判断中间推理步骤的正确性，结合PRM和QVM的Q方法，在基于相同LLM的各种方法中取得了最佳性能，使Llama-2-7b超越了源代码封闭的ChatGPT-turbo，准确率达到了80.8%。

Table 2: Comparison of Q* and other baselines on GSM8K dataset.

Base Model	SFT	Alignment	Verification	Accuracy
GPT-3.5 (5-shot) [50]	Unknown	PPO (RM) [31]	-	57.1%
ChatGPT-instruct (0-shot) [46]	Unknown	PPO (RM) [31]	-	71.3%
ChatGPT-turbo (0-shot) [46]	Unknown	PPO (RM) [31]	-	77.7%
GPT-4 (0-shot) [46]	Unknown	PPO (RM) [31]	-	91.9%
GPT-4 (5-shot) [50]	Unknown	PPO (RM) [31]	-	**92.0%**
Llama-2-7b (0-shot)	-	-	-	49.5%
Llama-2-7b (0-shot)	MetaMath[5]	-	-	65.2%
Llama-2-7b (0-shot)	MetaMath[5]	PPO (PRM) [31]	-	67.2%
Llama-2-7b (0-shot)	MetaMath[5]	PPO (QVM) [31]	-	67.6%
Llama-2-7b (0-shot)	MetaMath[5]	-	Best-of-N (PRM) [22]	72.1%
Llama-2-7b (0-shot)	MetaMath[5]	-	Best-of-N (QVM) [22]	74.5%
Llama-2-7b (0-shot)	MetaMath[5]	-	Q* (QVM)	78.8%
Llama-2-7b (0-shot)	MetaMath[5]	-	Q* (PRM+QVM)	**80.8%**

MATH。

通过查看表的结果，我们发现在MATH数据集上，MetaMath微调后的Llama-2-7b的性能较弱。因此，我们评估了另外两个更强的LLM模型来验证我们的Q方法的有效性。一个是经过合成数据微

都能进一步提高性能，超过了N选一方法。值得注意的是，经过Q增强的DeepSeek-Math-7b的性能已经超过了MATH数据集排行榜上的多个封闭式模型，如Gemini Ultra（4-shot），准确率达到了55.4%。

Table 3: Comparison of Q* and other baselines on MATH dataset.

Base Model	SFT	Alignment	Verification	Accuracy
GPT-3.5 (0-shot) [48]	Unknown	PPO (RM) [31]	-	23.5%
GPT-4 (0-shot) [48]	Unknown	PPO (RM) [31]	-	42.5%
Gemini Ultra (4-shot) [51]	Unknown	PPO (RM) [31]	-	**53.2%**
Llama-2-7b (0-shot)	-	-	-	7.9%
Llama-2-7b (0-shot)	MetaMath[5]	-	-	20.0%
Llama-2-7b (0-shot)	Synthetic Data[47]	-	-	41.9%
Llama-2-7b (0-shot)	Synthetic Data[47]	PPO (QVM) [31]	-	42.5%
Llama-2-7b (0-shot)	Synthetic Data[47]	-	Best-of-N (QVM) [22]	46.8%
Llama-2-7b (0-shot)	Synthetic Data[47]	-	Q* (QVM)	**49.1%**
DeepSeek-Math-7b (0-shot)	Unknown	PPO (QVM) [31]	-	50.8%
DeepSeek-Math-7b (0-shot)	Unknown	PPO (QVM) [31]	Best-of-N (QVM) [22]	54.3%
DeepSeek-Math-7b (0-shot)	Unknown	PPO (QVM) [31]	Q* (QVM) [22]	**55.4%**

MBPP。

对于MBPP数据集的比较，我们选择了开源LLM中最强的代码生成模型CodeQwen1.5-7b-Chat作为基准模型。我们训练了一个Q值模型（QVM）来估计Q值，并手动构建了一个用途函数来评估生成的代码。从结果中可以看出，Q方法在代码生成方面超过了N选一方法，在MBPP数据集上达到了77.0%的准确率，这在MBPP排行榜中是非常好的成绩。

Table 4: Comparison of Q* and other baselines on MBPP dataset.

Base Model	SFT	Alignment	Verification	Accuracy
GPT-3.5 Turbo (self-debug) [52]	Unknown	PPO (RM) [31]	-	72.8%
GPT-4 (self-debug) [52]	Unknown	PPO (RM) [31]	-	**80.2%**
CodeQwen1.5-7b-Chat (0-shot)	Unknown	PPO (QVM) [31]	-	74.6%
CodeQwen1.5-7b-Chat (0-shot)	Unknown	PPO (QVM) [31]	Best-of-N (QVM) [22]	75.0%
CodeQwen1.5-7b-Chat (0-shot)	Unknown	PPO (QVM) [31]	Q* (PRM+QVM) [22]	**77.0%**

原文《Q*: Improving Multi-step Reasoning for LLMs with Deliberative Planning》

发布于 2024-07-18 10:28 · IP 属地北京

OpenAI 大模型 大语言模型