以 seller 环境为例的 self-judge pipeline

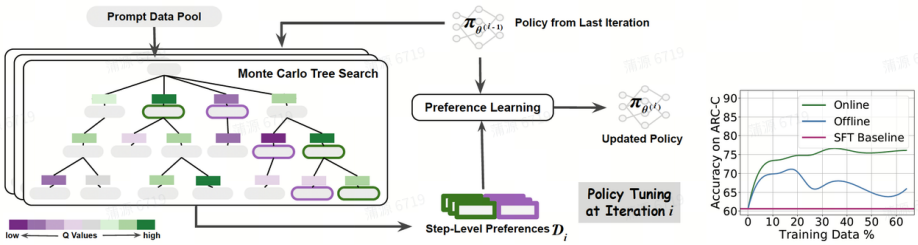
1. 动机

> - 李沐老师知乎：https://zhuanlan.zhihu.com/p/714533901
> - 愿景：人类陪伴和助手
>
> 这些综合在一起，我们把愿景定成了“**人类陪伴的智能体**”。一个**情商很高的，智商在线**的智能体。算换成现实中的人的话，应该会是一个专业团队。例如你想让它陪你玩，那它是**专业策划+演员**。陪你运动，那么**鼓励师+专业运动教练**。陪你学习，那么能把你不懂的讲懂。模型的好处是，它能做长期的陪伴，真的了解你。而且可以“真心为你”。

- 预训练得到的基础模型能力很重要，但我们主要专注于后训练，即向基础模型中注入期望的行为（例如感染力）

候选方案

1. 通过 SFT，将通用模型朝着我们希望的分布逼近
2. 从网络上收集大规模的有偏好的数据集，通过 DPO 微调
    - 对于不同子领域（例如直播销售，教学文章，演讲等），给出不同的偏好数据集
3. 合成数据（可以迭代的进行）
    - 通过 MCTS 生成合成的偏好数据，通过 DPO 微调
        - MCTS-DPO：结果正确性 (outcome correness) O 和自我评估 (self-evaluation) C。

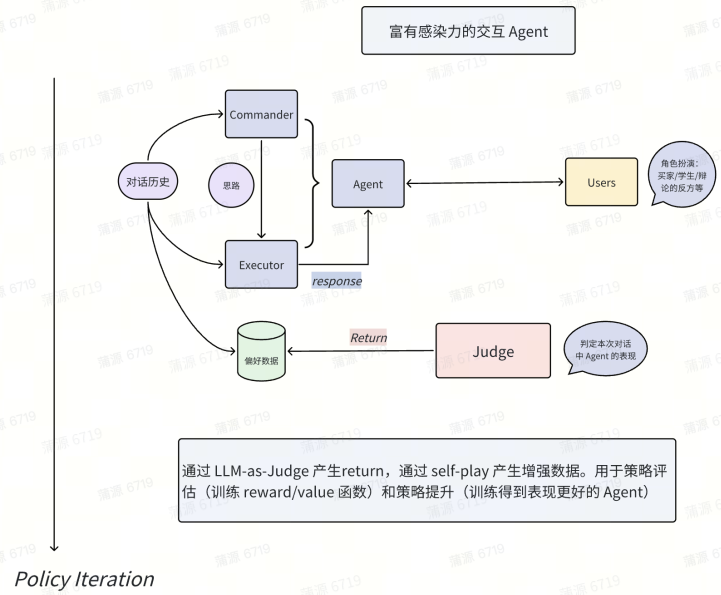


    - AgentQ：原理与MCTS-DPO类似
4. 层次化 RL 方法训练 Language Agent
    - ArCHer: Training Language Model Agents via Hierarchical Multi-Turn RL
5. 通过 self-play self-judge 来合成数据，通过上面的各种方法来提升模型的（推理等）性能

2. 方案

- 背景：目前通过 **SFT**，将通用模型朝着我们希望的分布逼近
  a. 目前主要专注单人的讲话过程，缺乏多轮对话过程过程
  b. 由于误差累积和有限的探索数据，可能在**多轮对话**任务上表现不佳
- 目的：多轮对话类似一个 self-play 的过程，如何增强多论对话的一致性和感染力？
  - 参考人类在（例如直播销售，讲课，辩论等）的思维过程
  - 验证是否能通过 MCTS 先显式地搜索一个讲话思路，然后再让模型按照思路输出响应

分层 LM 的架构设置



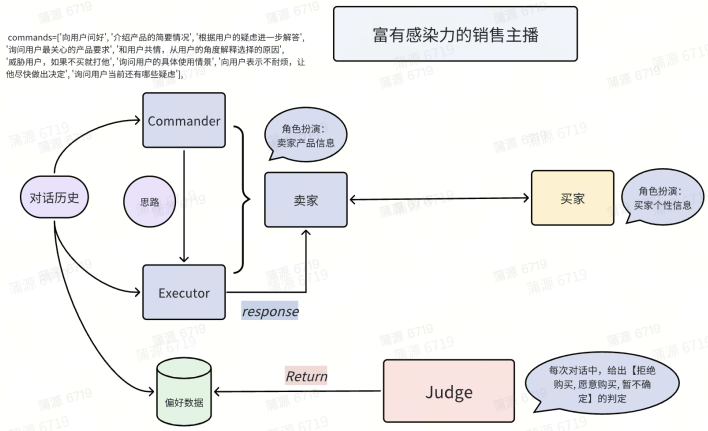
（图1：基于分层 LM 的多轮对话 Agent 策略提升示意图。）

- Agent 与 User 之间通过 self_play 的方式，产生数据与训练
- Agent
  - 上层 LM （Commander）根据**当前对话历史**给出 meta-action/command/thoughts $a_{meta} = \pi_u(x)$
    - 不同的任务，meta-atcion 如何定义？
      - LM 基于对话历史给出
      - 针对领域信息的预定义动作空间
  - 下层 LM （Excecutor）根据**当前对话历史和 command** 给出对话片段 $a = \pi_l(x, a_{meta})$
    - 如何保证下层 LM 有着严格遵循 a_{meta} 的能力
      - 基础模型的指令遵循能力
- Users
  - 由 LM 扮演

- Judge
  - 核心：需要有对整个文段的 reward 模型，才能进行引导 MCTS 搜索
  - 另一个 LM 作为 Judge 对整个多轮对话过程给出 reward

验证环境

- seller_env.py



（图2：以 seller 环境为例的 Agent 训练示意图。）

- **角色说明**
  - commander：其实就是需要学习的 policy，要生成上层的指令。是一个离散动作空间，初步的设计有9个动作：['向用户问好', '介绍产品的简要情况', '根据用户的疑虑进一步解答', '询问用户最关心的产品要求', '和用户共情，从用户的角度解释选择的原因', '威胁用户，如果不买就打他', '询问用户的具体使用情景', '向用户表示不耐烦，让他尽快做出决定', '询问用户当前还有哪些疑虑']
  - executer：即真正的卖家发言，根据 commander 提供的思路做出具体的说辞
  - buyer：买家，根据历史 commander 和 buyer 的信息假装买家进行回复
  - judge：评判者，评判环境是否结束，买家是否有很大的概率购买
- **动作空间和观察空间**
  - 动作空间：commander 可以选择的上层策略，目前包含七个动作
  - 观察空间：executer 和 buyer 对话的所有历史记录
  - 奖励函数：卖家能不能在一定的对话轮次数量内，让 judge 给出 “愿意购买” 的判定。如果能就是 +1，否则为 -1（稀疏奖励设计）
- **环境 step**
  - 根据传入的 action id 得到具体的上层指令
  - Executer 按照上层指令和历史对话信息生成新的推销话术，更新历史对话信息
  - Buyer 根据所有历史对话信息，进行进一步回复，更新历史对话信息

- Judge 根据历史对话信息，评判环境的奖励函数

3. 现有代码测试

- **代码链接**：https://github.com/opendilab/LightZero/pull/276
- **状态**：
  - MCTS bot 在一个 demo env（只有4个动作，每个动作的奖励是固定值）上验证收敛
  - AlphaZero 在最简易场景上（单个商品多人设）验证收敛，但在多商品多人设上性能波动，可能是由于LM扮演Buyer和Judge本身具有随机性
- **代码结构**
  - 在本地部署基础 llm 服务（model_serve）
    - 具体参考：https://lmdeploy.readthedocs.io/zh-cn/latest/llm/api_server.html
    - pip install lmdeploy
    - lmdeploy serve api_server Qwen/Qwen2-7B-Instruct --server-port 23333
    - zoo/seller/utils.py: self.agent == 'lmdeploy'中的 base_url 替换 lmdeploy实际的url
  - seller_env: zoo/seller/seller_env.py
  - mcts bot : zoo/seller/seller_mcts_bot.py
  - alphazero 训练入口: zoo/seller/config/seller_alphazero_config.py

4. 待解决问题

- Agent 上层 LM （Commander）动作空间是很大的
  - 预定义动作空间：如果固定相当于人为设置了性能上限
  - 如果不固定，如何处理变化的动作空间: action_mask
- Users 模型拟人的真实性
  - 提示词优化
- Judge 模型的准确性
  - Judge 给出的 reward 信息决定着优化的方向
    - **可以先通过偏好数据 SFT LM 得到**
- 多轮对话之间的连贯性
  - 目前没有强制约束来优化连贯性
  - 可以使用代理指标
- meta-atcion 的粒度

- 一个 meta-atcion 不一定对应一轮