AgentEvolver

高效自进化Agent系统

Towards Efficient Self-Evolving Agent System

研究机构

阿里巴巴通义实验室

发布日期

2025年11月13日

资源链接

论文链接

GitHub仓库

课程章节导航

核心亮点

- 首个实现Agent完全自主学习的系统框架
- 无需人工标注即可生成高质量训练数据
- 显著提升RL探索效率和样本利用率
- 在AppWorld和BFCL v3基准测试中取得SOTA效果

↑ 返回顶部

核心贡献

核心痛点

当前基于 LLM 的 Agent 开发面临两大难题:

1. **数据构建成本极高**
   在新环境中缺乏现成的高质量任务数据
2. **探索效率低下**
   传统强化学习探索策略在长序列任务中样本效率极低

核心方案

论文提出了一个名为 **AgentEvolver** 的自进化Agent系统,该系统通过三大协同机制实现持续、高效的自我迭代与能力演进:

❓ 自我提问

Self-Questioning

利用好奇心驱动探索并自动合成训练任务

🧭 自我导航

Self-Navigating

通过经验复用和混合策略引导探索

🎓 自我归因

Self-Attributing

利用LLM进行细粒度的过程奖励分配

主要贡献

1 Self-Questioning (自提问)

利用好奇心驱动探索未知环境并自动合成训练任务,解决了冷启动数据匮乏问题

2 Self-Navigating (自导航)

通过经验复用和混合策略引导,显著提升了RL的探索效率

3 Self-Attributing (自归因)

利用LLM进行细粒度的过程奖励分配,解决了长序列任务中稀疏奖励导致的低样本效率问题

4 SOTA性能

构建了标准化的Agent训练基础设施,并在AppWorld和BFCL v3基准测试中,用较小参数模型(14B / 7B)取得了超越大模型的SOTA效果

实验结果

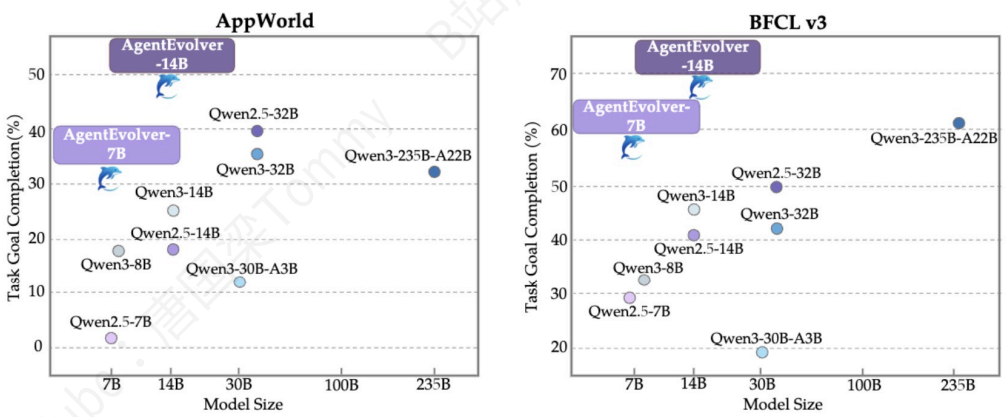

Figure 1: Performance comparison on the AppWorld and BFCL-v3 benchmarks. AgentEvolver achieves superior results while using substantially fewer parameters than larger baseline models.

图1: AgentEvolver在AppWorld和BFCL v3基准测试上的性能对比

图示说明: 图1展示了AgentEvolver在AppWorld和BFCL v3基准测试上的性能对比结果。实验结果表明,AgentEvolver使用较少的参数(14B/7B)即可取得超越大模型(如235B参数模型)的SOTA效果,充分证明了其高效自进化机制的有效性。

↑ 返回顶部

研究背景

现有范式

当前,Agent的开发主要遵循两种范式:

1 工作流编排

Workflow-based Orchestration

开发者手动设计固定的动作序列或工具调用规则,Agent像一个脚本执行器,**灵活性和适应性差**。

⚠ 缺点: 难以应对复杂多变的场景

2 基于学习的优化

Learning-based Optimization

主要采用强化学习(RL),尤其是PPO/GRPO等策略优化算法,让Agent通过与环境的交互和反馈来学习策略。

✓ 优点: 更有潜力实现智能化

关键痛点

基于RL的方法虽然更有潜力,但面临**三大挑战**:

1 任务构建成本高

在全新的环境中,需要人工设计大量多样化的训练任务,这个过程**既耗时又费力**。

问题: 缺乏现成的高质量(Instruction, Trajectory)数据对

2 探索效率低下

Agent通常依赖大规模的**随机试错**来探索环境,产生大量冗余且无学习价值的交互轨迹,样本效率极低。

问题: 在长序列任务中如同大海捞针,难以命中成功轨迹

3 信用分配不精准

在长序列任务中,传统的RL方法通常给予整个轨迹一个统一的奖励,**无法区分哪些步骤是关键决策**,哪些是无效操作,导致学习信号模糊。

问题: 稀疏奖励导致样本效率极低,收敛缓慢

研究空缺

核心研究问题

论文试图填补现有研究中**"如何让Agent自主、高效地驱动自身学习过程"**这一空白。

它挑战了传统依赖 "外部人类工程师" 设计学习流程的范式,提出了一种由 "Agent内部LLM" 主导学习循环的新范式,旨在实现更具扩展性、成本效益和持续性的Agent能力进化。

AgentEvolver的创新突破

?

自我提问

解决任务构建成本高的问题

🧭

自我导航

解决探索效率低下的问题

🎓

自我归因

解决信用分配不精准的问题

↑ 返回顶部

核心思想

整体框架图

AgentEvolver 建立了一个自训练闭环,遵循 "环境 → 任务 → 轨迹 → 策略" 的演进路径。系统不仅学习怎么做(Policy),还负责生成学什么(Task)以及如何评价(Reward)。

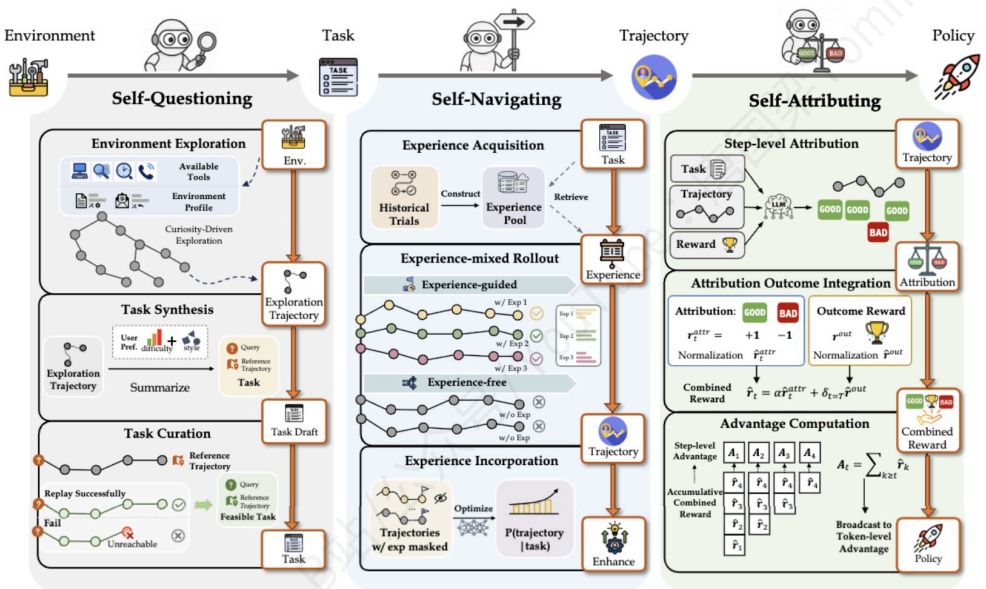

Figure 2: Overview of the AgentEvolver framework. The self-evolving process is driven by three synergistic mechanisms: **Self-questioning** for autonomous task generation, **Self-navigating** for experience-guided exploration, and **Self-attributing** for fine-grained credit assignment.

图2: AgentEvolver自训练闭环 - 展示了环境、任务、轨迹、策略之间的循环演进关系

数据流转过程

1 环境 → 任务 (Self-Questioning)

Agent首先探索环境,然后基于探索的发现和用户偏好,自主地生成一系列可行的任务。

- 🎯 **核心机制:**好奇心驱动的逆向任务生成
- 💡 **输出:**合成的训练任务及其参考解

2 任务 → 轨迹 (Self-Navigating)

面对生成的任务,Agent利用一个"经验池"(从过去的成功和失败中总结的知识)来指导自己的行动,生成更高质量的解决方案轨迹,而不仅仅是盲目试错。

- 🗂️ **核心机制:**经验复用 + 混合策略引导
- 📈 **效果:**大幅提升探索效率和成功率

3 轨迹 → 策略 (Self-Attributing)

在生成轨迹后,Agent回顾整个过程,对每一步行动进行"好"或"坏"的评价,形成一个细粒度的奖励信号。这个信号随后被用来更精确地更新和优化Agent的决策策略。

- 🔍 **核心机制:**逐步归因 + 复合奖励融合
- ⚡ **优势:**精准的信用分配,2-3倍样本效率提升

🔄

自我强化循环

这个优化的策略将用于新一轮的环境探索,形成一个不断自我强化的循环

各模块协作

📋 Context Manager

负责维护多轮交互的上下文窗口,确保Agent能够理解和记住历史信息

📝 Task Manager

负责生成和过滤任务,确保任务的质量和可行性

🗂️ Experience Manager

负责RAG式的经验检索,为Agent提供相关的历史经验指导

🏗️ Infrastructure

实现了Gym兼容的并行环境执行,提供高效的训练基础设施

核心理念总结

AgentEvolver的核心理念是让Agent从一个 "被动的学习者" 变成一个 "主动的自我进化者" 。通过三大机制的协同工作,Agent不仅能够学习"怎么做",还能自主决定"学什么"和"如何评价",从而实现真正的自主学习和持续进化。

↑ 返回顶部