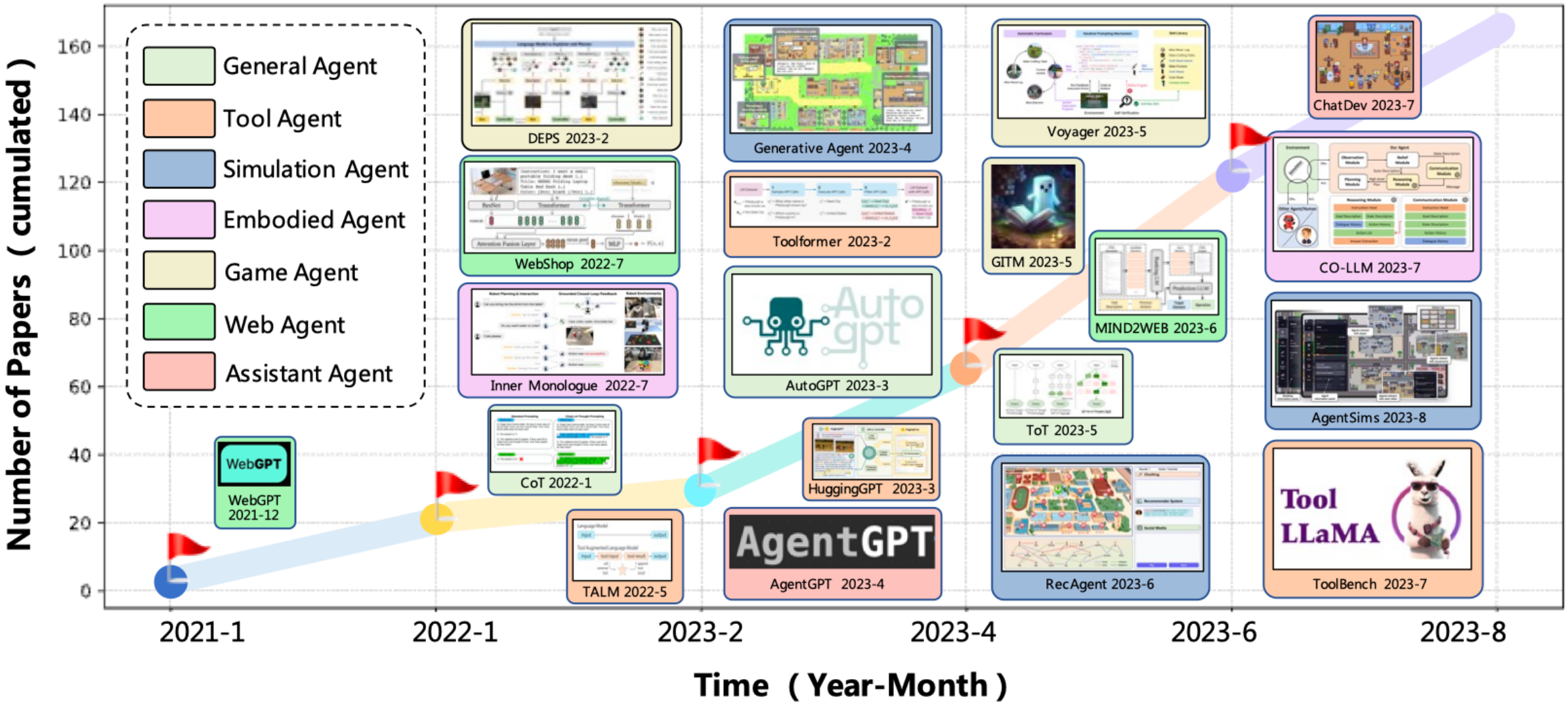
大模型系列之智能体

Agent 问题与挑战

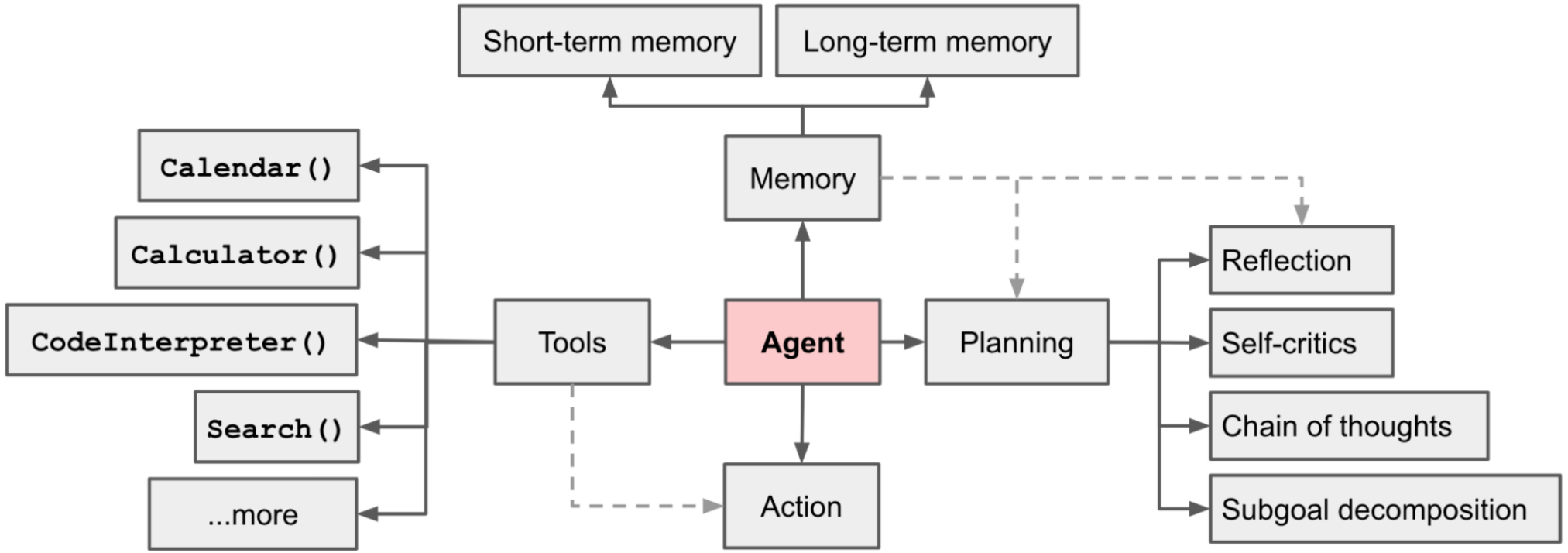
关于大模型系列

- 内容背景
  - LLM + AI Agent：大模型遇到智能体
- 具体内容
  1. AI Agent 组成介绍：LLM + 记忆 + 规划 + 工具
  2. AI Agent 规划手段： Task Decomposition 与 Self Reflection
  3. AI Agent 热门应用：交互式 Agent、自动化 Agent 与多模态 Agent
  4. AI Agent 问题与挑战： Agent 的问题、Agent 的局限性


General Agent
Tool Agent
Simulation Agent
Embodied Agent
Game Agent
Web Agent
Assistant Agent
Number of Papers (cumulated)
0
20
40
60
80
100
120
140
160
2021-1
2022-1
2023-2
2023-4
2023-6
2023-8
Time (Year-Month)
WebGPT
WebGPT 2021-12
DEPS 2023-2
WebShop 2022-7
Inner Monologue 2022-7
CoT 2022-1
TALM 2022-5
Generative Agent 2023-4
Toolformer 2023-2
AutoGPT 2023-3
HuggingGPT 2023-3
AgentGPT
AgentGPT 2023-4
Voyager 2023-5
GITM 2023-5
ToT 2023-5
RecAgent 2023-6
MIND2WEB 2023-6
ChatDev 2023-7
CO-LLM 2023-7
AgentSims 2023-8
Tool LLaMA
ToolBench 2023-7

关键组成

- 规划 Planning + 记忆 Memory + 工具 Tools



1. 问题挑战

局限性1：效率

- **效率**

1. **执行效率**：Agent 作为执行者，需要运行多次和与外界进行交互，LLMs 消耗资源很大，因此导致 Agents 运行效率不会太高。
2. **探索效率**：通过 Agent 自行探索并完成整个解决过程仍然比较繁琐耗时，Agent 也很容易把问题复杂化。考虑到 LLM 调用的成本，要在实际场景落地使用也还需要在这方面做不少优化。一种方式可能是像 AutoGPT 那样可以中途引入人工的判断干预和反馈输入。

局限性II：依赖 LLM

- **依赖LLM：**Agent 所有的衍生技术都严重依赖于 LLMs 基础能力（OpenAI GPT4 API），所以一个 Agents 的好坏，需要看该 LLMs 的训练 or 微调学习是否到位。

1. **动作有效性。**评估过程中， LLMs 并不总是在遵循指令。即 LLMs 预期输出并不总是处于环境可以接受的输入空间中。如 1）LLMs 没有理解指令，因此没有输出动作；2）LLMs 输出动作，却是错误或不完整。所以如何确保动作有效，是一个需要改进的方向。
2. **长上下文。**开源模型的上下文长度（如2k tokens）会极大地影响交互任务中的表现，有些任务需要较长的指令和反馈，可能超过上下文长度，导致模型忽略有用信息。
3. **多轮一致性，不稳定性。**部分任务需要多轮对话，每轮对话简短。会导致 LLMs 在多轮对话中丢失自我角色（如输出道歉并表示无法回答）。在多轮对话 or 任务中保持一致性，是一个具有挑战性的工作。

局限性III：训练方式和效果

- **记忆召回**：如果只是做简单的 embedding 相似性召回，很容易发现召回的结果不是很好。这里应该也有不少可以改进的空间，Generative Agents 里对于记忆的更细致的处理，LlamaIndex 中对于 index 结构的设计也有很多可以选择与调优的地方。
- **错误累积**：实际上 AI Agent 总体表现并没有那么惊艳，网上 or 论文给出的很多例子都都是经过精挑细算。真正可能会因为前面一些步骤出现偏差，后续的执行步骤逐渐越跑越远。很重要的问题可能还是任务规划 Planning and 执行 Action，外部工具利用等方面的高质量训练数据相对匮乏。这或许是 OpenAI 自己做 Tools plugin 体系的原因之一。

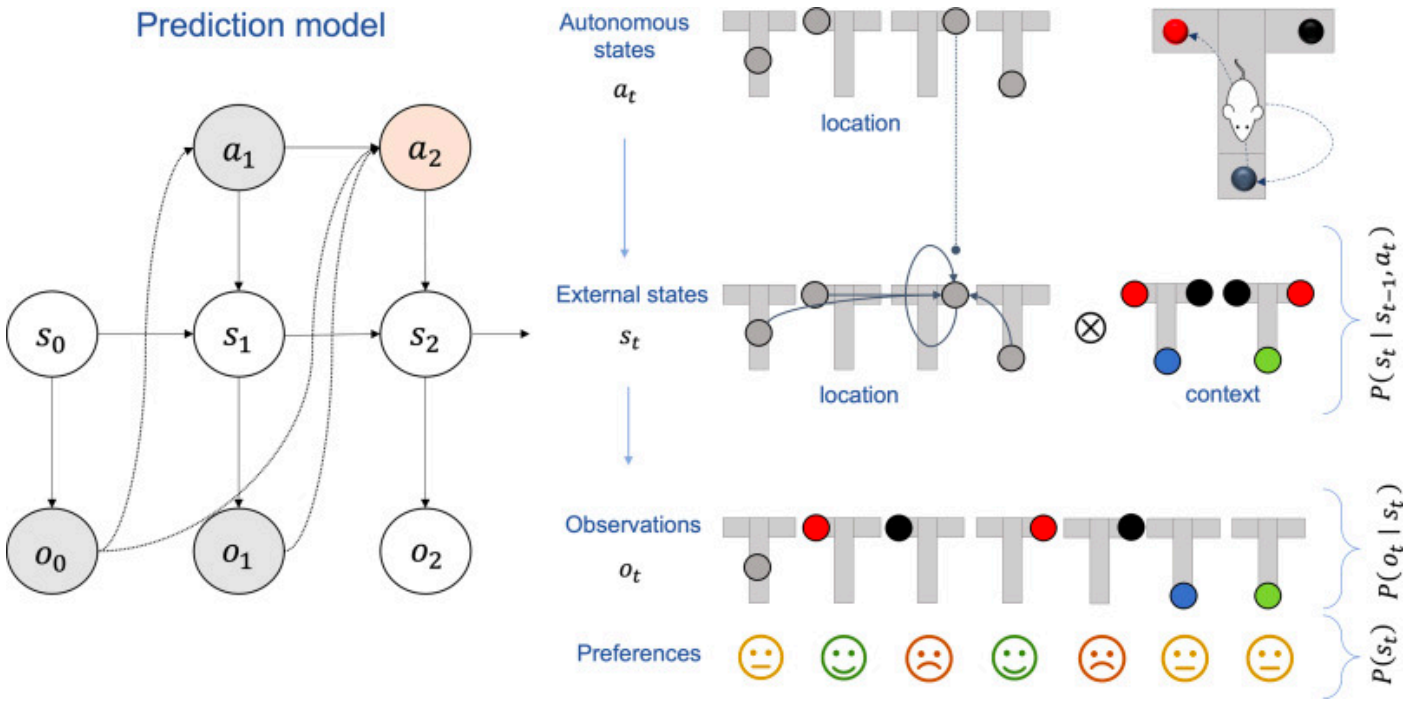
2. 思考

有监督任务

- 对于 Classification、 Regression 的监督 Supervised Learning 任务，智能体生成一步即决定能否完成任务。
- **结论**：监督任务中一次执行即完成的智能体 Agent 与规划 Planning 无关，并不是目前所讨论的 Agent。

序列决策任务

- 对于 Sequential decision making 的多步序列决策任务来说，每一步执行的决策，会存在每一个 Action 都可能影响以后的决策，最终很可能影响整个决策的质量。一步步生成 Action 很有可能导致智能体的决策不是最优解。



序列决策任务一定是 LLMs ？

- 以最短路径问题为例，Agent 如果每次根据目前的情况找最近的邻居节点，很有可能找不到最短路径。在某一步选错了下一个节点，可能导致最终找不到最短路径。
- 多步序列决策 Sequential decision making 需要通盘考虑，因此需要结合 Planning、 RL 等算法，制定 or 学习最优策略。但是目前大部分 LLMs 没有专门训练多步决策。
- AlphaGo 看似每次生成一个落子决定，其实，AlphaGo在下棋之前，已经通过RL、MCTS等学好了很强的通盘的策略。因此看多步决策或许也不一定用 LLMs。目前 LLMs 还不能根据棋谱信息很好地下棋。

回顾历史 Agent

- Pre-training or Fine-tuning 阶段实现 Language Agent 与利用 Prompt engineering 实现 Language agent，属于两个完全不同的研发路径，虽然从表面上看起来，都在做language agent。
  1. Google DeepMind 创建的时候主攻 Autonomous Agents，如 AlphaGo、AlphaStat 等。
  2. OpenAI 主攻从预训练到 RLHF 的 Autonomous Agents，如发布从 GPT1 到 GPT4 。

Question ?

1. **GAI 是否只是个理想？**目前从技术演进方向看，有多条路径可以通向不通子领域的GAI，未来 GAI 仍然是Model Based。
2. **如何评估 AI Agent？**如果脱离外部环境，只是跟人闲聊天，那到底聊得好不好，是个很主观的事情，机器难以评判，人也不容易评判。

Thank you

把AI系统带入每个开发者、每个家庭、每个组织，构建万物互联的智能世界

Bring AI System to every person, home and organization for a fully connected, intelligent world.

ZOMI

Course chenzomi12.github.io

GitHub github.com/chenzomi12/DeepLearningSystem