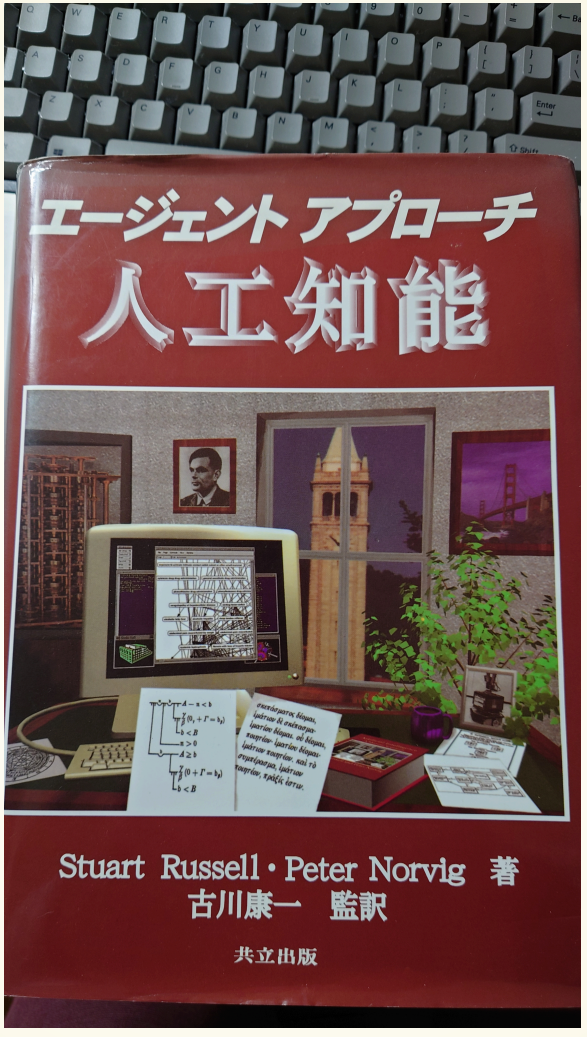
最近話題のAgentについてちょっと調べてみた

SciPyData 2025
1/25, 2025
辻真吾（www.tsjshg.info）

エージェントアプローチ 人工知能（1997年出版）



共立出版から第2版が2008年にでてます

エージェントとは？

本書のねらいは，ある環境で「うまくやる」エージェントを設計することである．まず「うまくやる」が何を意味するかを定義し，それからうまくやるエージェントを設計するためのいくつかの方法を述べる．（これは図 2.1 の疑問符の部分にあたる．）本書はエージェントの設計に用いられる一般的な方法論を述べるが，それはとりわけエージェントが何を知るべきかという原理である．最後に，エージェントを環境に適合させる方法と，何種類かの環境について述べる．

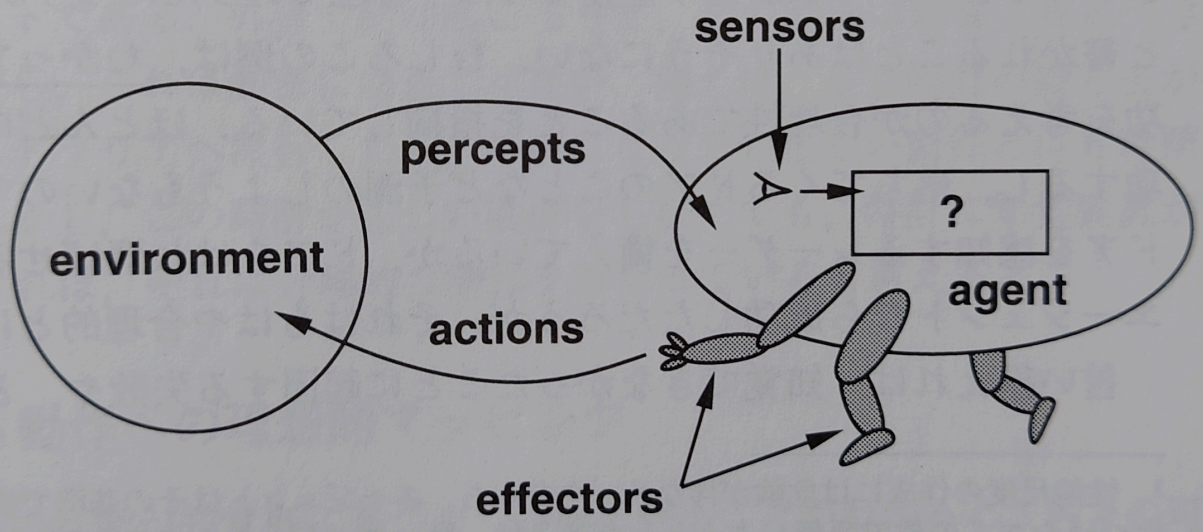


図 2.1　エージェントはセンサとエフェクタを介して環境に動作する．

いまどきのAIエージェント

学習済みのLLM（Larget Language Model）を使って作られたエージェントが想定され、複雑なタスクを自律して解決してくれることが期待される

- 推論（reasoing, 論理的な思考）
- 計画（planning）
- 実行（tool execution）

なぜエージェントが流行っているのか？（私見）

- 現在の大規模言語モデルは、潜在空間に埋め込まれたベクトル表現をもとに、文脈に応じてそれっぽいことを出力する機械
- Hallucination（幻覚）などの問題がある
- プロンプトエンジニアリング（Chain-of-Thoughtなど）やRAG（Retrieval-Augmented Generation）などの解決策が提案されている
- エージェント指向でさらなる改善を目指せないか？

2024/4に出た調査論文

The Landscape of Emerging AI Agent Architectures for Reasoning, Planning, and Tool Calling: A Survey

Tula Masterman*
Neudesic, an IBM Company
tula.masterman@neudesic.com

Sandi Besen*
IBM
sandi.besen@ibm.com

Mason Sawtell*
Neudesic, an IBM Company
mason.sawtell@neudesic.com

Alex Chao
Microsoft
achao@microsoft.com

* Denotes Equal Contribution

Abstract

This survey paper examines the recent advancements in AI agent implementations, with a focus on their ability to achieve complex goals that require enhanced reasoning, planning, and tool execution capabilities. The primary objectives of this work are to a) communicate the current capabilities and limitations of existing AI agent implementations, b) share insights gained from our observations of these systems in action, and c) suggest important considerations for future developments in AI agent design. We achieve this by providing overviews of single-agent and multi-agent architectures, identifying key patterns and divergences in design choices, and evaluating their overall impact on accomplishing a provided goal. Our contribution outlines key themes when selecting an agentic architecture, the impact of leadership on agent systems, agent communication styles, and key phases for planning, execution, and reflection that enable robust AI agent systems.

Keywords AI Agent · Agent Architecture · AI Reasoning · Planning · Tool Calling · Single Agent · Multi Agent · Agent Survey · LLM Agent · Autonomous Agent

https://arxiv.org/abs/2404.11584

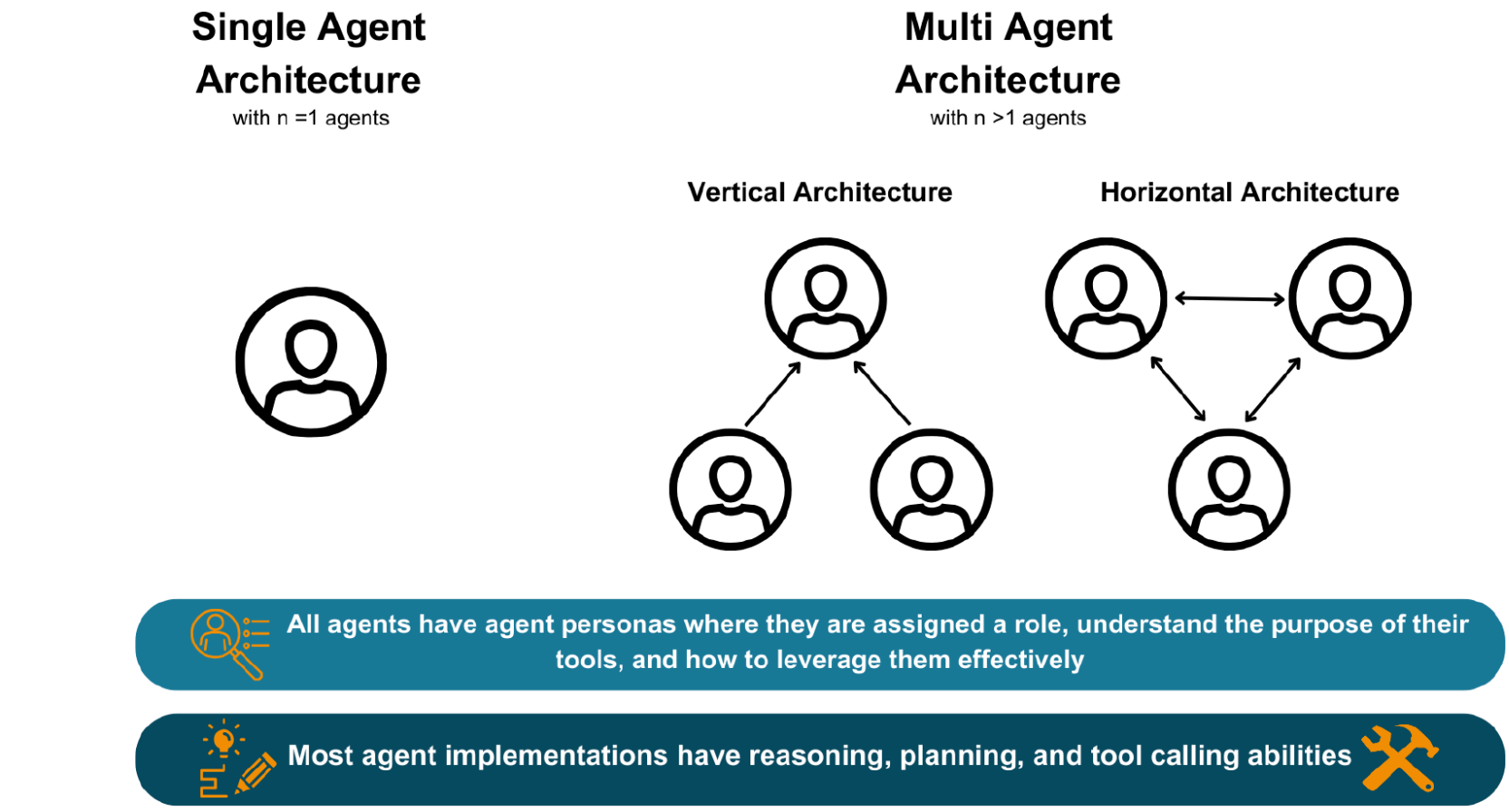


Figure 1: A visualization of single and multi-agent architectures with their underlying features and abilities

推論と計画の重要性

- 複雑な問題を正確に捉え、それを解くためには正しい推論が必要（人でも同じこと）
- 推論したあとに計画を作る必要がある
  - タスクの分割（task decomposition）、multi-plan selection, external module-aided planning, reflection and refinement, memory-augmented planningなどなど
- Plan Like a Graph（PLaG）
  - 計画を有向グラフで表現する方法
  - 性能向上に寄与するが、ステップ数が増えると性能が低下する

Graph-enhanced Large Language Models in Asynchronous Plan Reasoning

Fangru Lin [1] **Emanuele La Malfa** [1 2] **Valentin Hofmann** [1 3 4] **Elle Michelle Yang** [1]
Anthony G. Cohn [2 5] **Janet B. Pierrehumbert** [1]

Abstract

Planning is a fundamental property of human intelligence. Reasoning about asynchronous plans is challenging since it requires sequential and parallel planning to optimize time costs. Can large language models (LLMs) succeed at this task? Here, we present the first large-scale study investigating this question. We find that a representative set of closed and open-source LLMs, including GPT-4 and LLaMA-2, behave poorly when not supplied with illustrations about the task-solving process in our benchmark AsyncHow. We propose a novel technique called *Plan Like a Graph* (PLaG) that combines graphs with natural language prompts and achieves state-of-the-art results. We show that although PLaG can boost model performance, LLMs still suffer from drastic degradation when task complexity increases, highlighting the limits of utilizing LLMs for simulating digital devices. We see our study as an exciting step towards using LLMs as efficient autonomous agents. Our code and data are available at https://github.com/fangru-lin/graph-llm-asynchow-plan.

1. Introduction

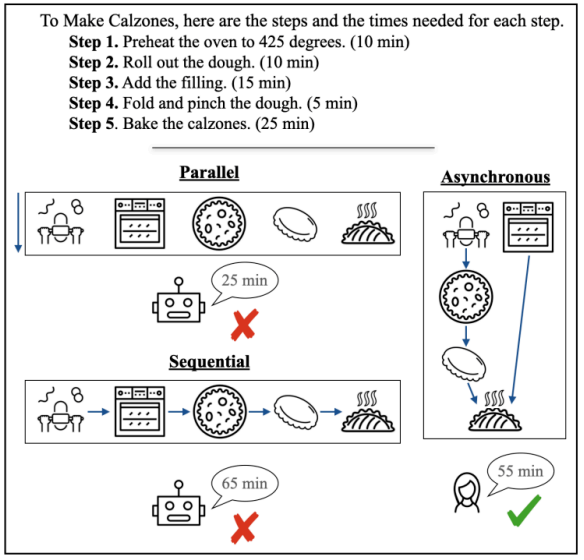


Figure 1. A planning task (top) can be executed sequentially, in parallel, or asynchronously. Blue arrows denote action ordering constraints. Although complete parallelism is logically the most time-efficient strategy, it results in invalid reasoning steps (e.g. `'Baking'` cannot happen at the same time with `'Rolling the dough'`); at the same time, sequentially executing each task negatively affects efficiency. Given infinite resources, an optimal (asynchronous) plan should parallelize actions wherever possible.

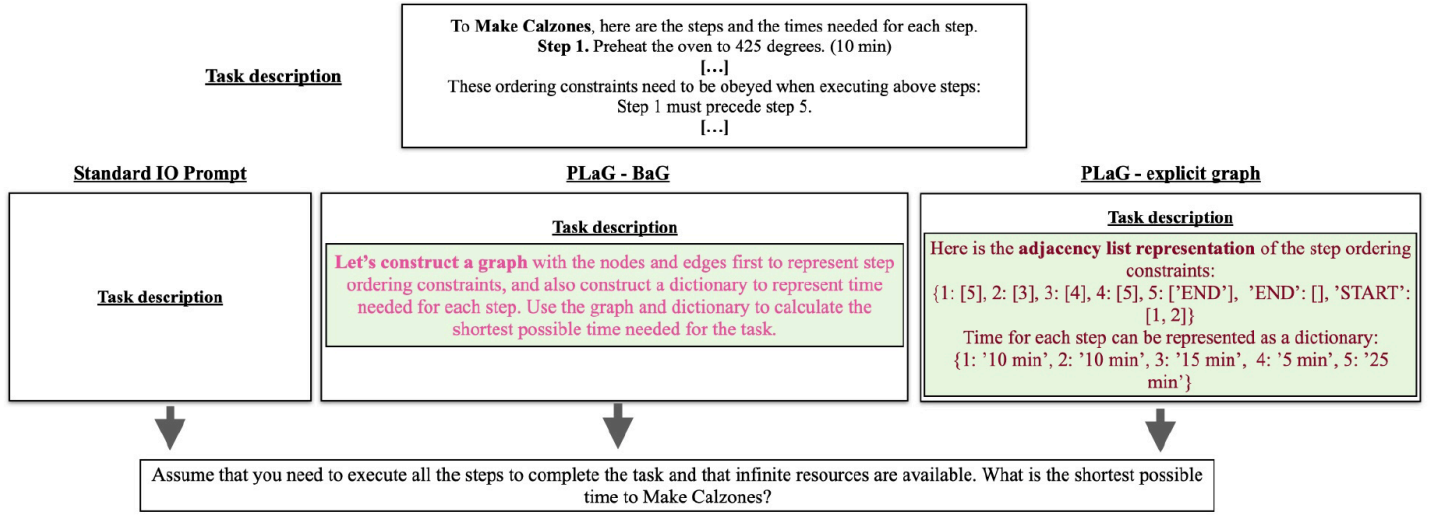


Figure 2. Comparing standard Input-Output (IO) prompting with our method (PLaG). Here, we illustrate PLaG (explicit graph) with an adjacency list, but it can be of any graph type in practice. The standard IO method is similarly deployed in zero-shot, zero-shot + CoT, *k*-shot, *k*-shot + CoT in this paper. Please refer to Appendix A.8 for more details.

- タスクを達成するのにかかる最小の時間を答えさせる
- PLaG - explicit graphはグラフ構造を隣接リストの構造で明示
- PLaG - BaGは説明の中で隣接リストを明示して、こういう感じのグラフを作ってから計算しようと指示する

結果1

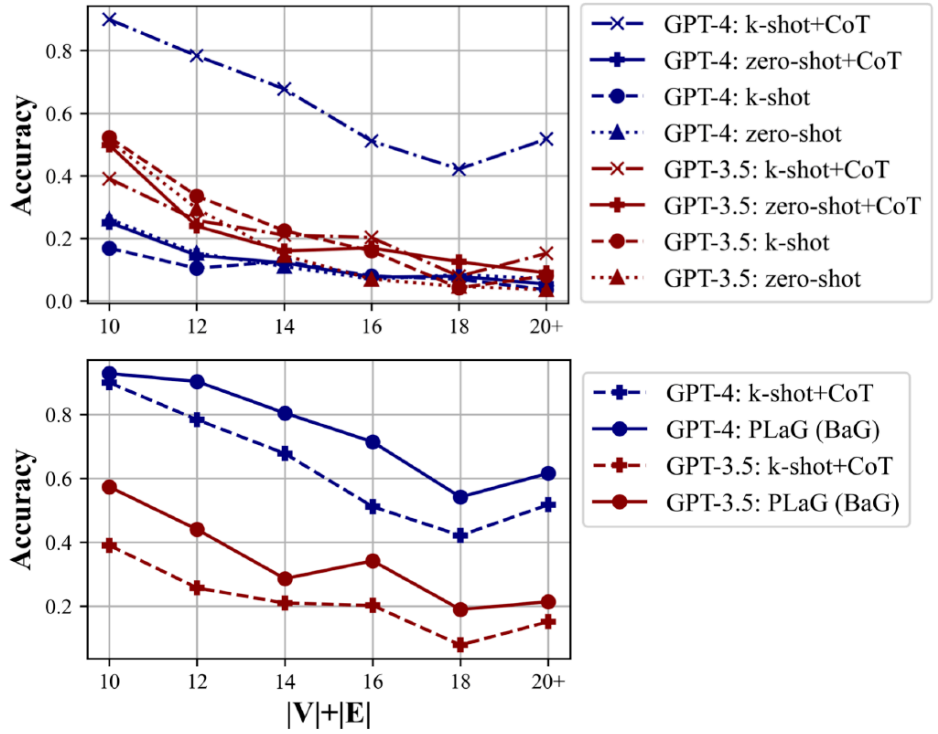


Figure 3. GPT-3.5 and GPT-4 accuracy as a function of asynchronous planning task complexity $|V| + |E|$ (see Section 2), after binning results by width of 2. The upper figure plots the performance of methods without PLaG (our method), and the lower plot displays the best method with/without PLaG.

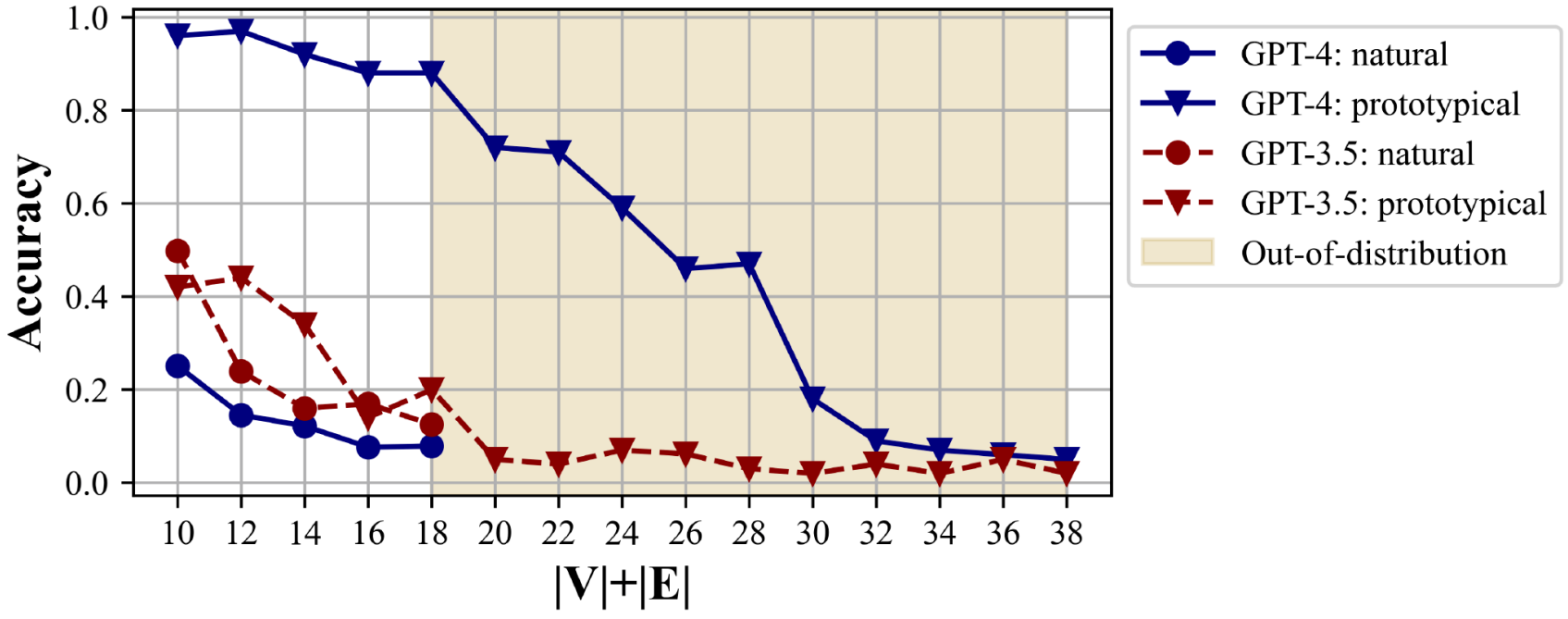
結果2

隣接リストを明示的に与えるより、説明文の中で紹介して自分でグラフを作らせるBaGの方が若干性能が良い

Table 2. Model accuracy in different settings on the AsyncHow benchmark. Model performances without our method are in plain background, while those with our method are in blue background. We mark the best performance per model in **bold**. Following Dror et al. (2018), we use McNemar's tests (McNemar, 1947) to obtain p-values and Holm-Bonferroni method (Holm, 1979) to correct them for each evaluation to test the statistical significance of performance difference between experiment with and without our proposed method. We denote with † when the performances with PLaG are significantly better ($p < 0.05$) than the best result without.

	Without PLaG				With PLaG	
Model	zero-shot	zero-shot + CoT	k-shot	k-shot + CoT	PLaG (explicit graph)	PLaG (BaG)
GPT-4	0.130	0.129	0.107	0.657	0.730†	**0.777**†
GPT-3.5	0.199	0.224	0.248	0.226	0.290†	**0.355**†
Command	0.078	0.015	0.050	0.078	**0.100**	0.050
LLaMA-2-70B-chat	0.039	0.038	0.053	0.076	**0.101**†	0.069
Mistral-7B-Instruct	0.078	0.070	0.098	0.149	**0.161**	0.146

結果3



手順が複雑になると急激に性能が低下する

結論

7. Conclusion

In this paper, we automatically generate a benchmark, AsyncHow, and assess LLMs for their performance in asynchronous plan reasoning. We find that if not provided with a detailed illustration of the task solution process, all models behave extremely poorly in our task. We propose a formalism to classify naturalistic asynchronous planning tasks, which successfully predicts LLMs' performance patterns. We propose PLaG, a method that consistently boosts SOTA model performance across all task complexity levels off the shelf. Despite this, we find that model performance still drastically downgrades with increasing task complexity, which calls into question using them as digital devices or generally intelligent agents.

LLMをデジタルデバイスや汎用的なエージェントに仕立てあげられるのか疑問

効果的なtool callingの重要性

- 問題を適切に分割して、分割された問題に適したAPIを呼ぶことで複雑な問題を解決する
- それぞれにペルソナを持ったマルチエージェントでは、うまく問題を分割できれば性能向上が見込める

シングルエージェント

- 複雑な問題の場合、うまく計画が立てられず、堂々巡りに陥ることがある。
- シングルエージェントには、よく定義されたわかりやすいタスクが向いている
- 提案されている主な手法
  - ReAct（Reason+Act）
  - RAISE
  - Reflexion
  - AUTOGPT+P
  - LATS

マルチエージェント

- チームの編成と改変(reorganization)に強みがある
  - それぞれのエージェントが個性に合わせた仕事に特化
  - 次のステップで必要なくなればチームから消す
- 提案されている主な手法
  - Embodied LLM Agents Learn to Cooperate in Organized Teams
  - DyLAN
  - AgentVerse
  - MetaGPT

Discussion and Observation

- プロンプトで与えられた情報が十分だった場合、マルチエージェントの議論がパフォーマンスの向上に寄与しないという研究もあるので、シングル・マルチの選択は広い視野が必要
- 人の正しいフィードバックはパフォーマンスを改善する
  - LLMベースのエージェントは意見に従順すぎる点は注意
- はっきりした役割設定はシングル・マルチエージェントともにパフォーマンスの向上に寄与
  - マルチエージェントではリーダーの役割をはっきり決めることも重要
- ほとんどの論文がよくあるベンチマーク用のデータセットを利用している
  - 現実世界にどれくらい役立つかはまだまだこれからの研究が必要