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

등반자팀
이진선

목적

- 기존 AI 에이전트 구현의 현재 능력과 한계를 전달

- 시스템이 실제로 작동하는 모습을 관찰하면서 얻은 통찰을 공유

- AI 에이전트 설계의 미래 발전을 위한 중요한 고려사항을 제시

서론

RAG(Retrieval Augmented Generation)

Agents

분류

Agents : 언어 모델 기반으로 계획하고 실행하며, 단일 또는 다중 구조로 구성

Agent Persona : 에이전트의 역할, 성격, 도구 사용 방식을 정의하며, 성능에 직접적 영향

Tools : 외부와 상호작용할 수 있게 해주는 함수로, 정보 조회·저장 등을 가능

Single Agent Architectures : 하나의 언어 모델이 모든 작업을 혼자 처리하며, 에이전트 간 피드백은 존재하지 않음

Multi-Agent Architectures : 여러 에이전트가 협력하여 작업하며, 모델·도구·역할을 분담

Vertical Architectures : 리더 에이전트가 중심이 되어 다른 에이전트들을 지휘·조율

Horizontal Architectures : 모든 에이전트가 동등하게 협업하며, 자율적으로 소통하고 행동

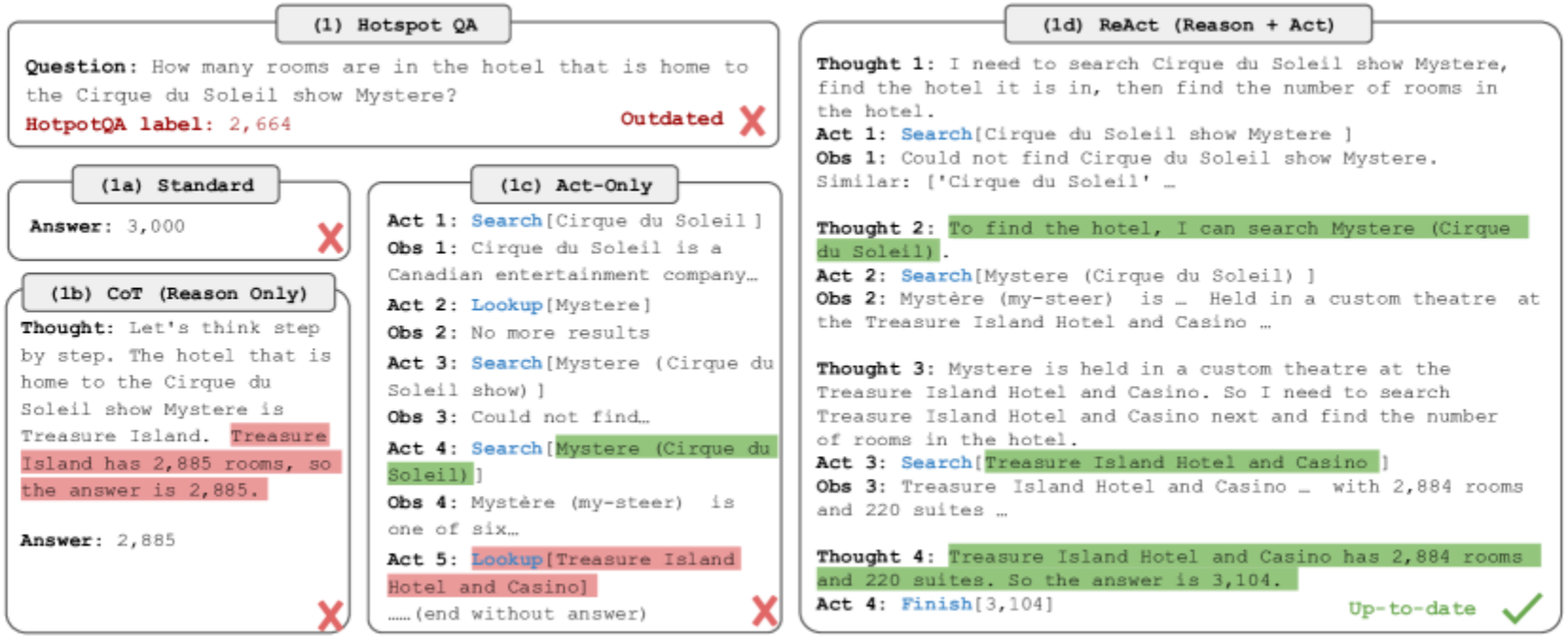
효과적인 에이전트를 위한 주요 고려 사항

- 에이전트의 개요
  에이전트는 언어 모델을 확장해 실제 문제를 해결하며, 문제 해결 능력, 추론과 계획 수립,
  외부 도구와의 상호작용이 필요

- 추론과 계획 수립의 중요성
  추론은 자율적 문제 해결에 필수적이며, 계획 수립은 작업을 효율적으로 나누고 선택하는 과정

- 도구 호출의 중요성
  에이전트는 외부 도구를 사용해 복잡한 문제를 해결하며, 다중 에이전트 시스템은 효율성 증가 기여

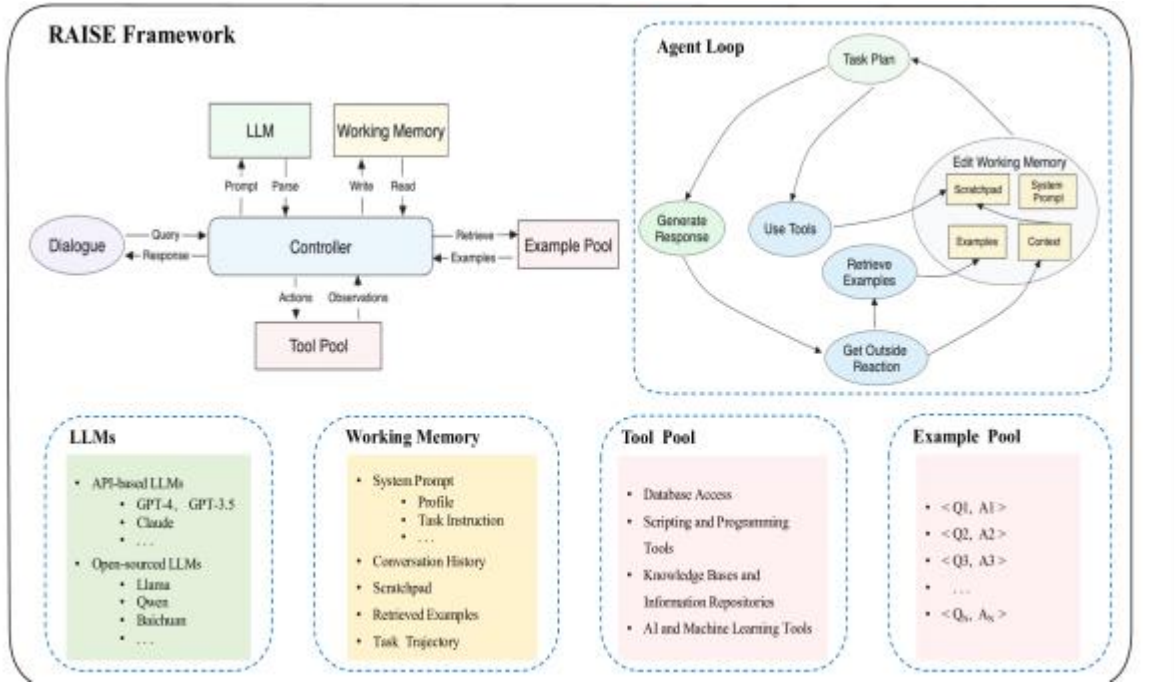
Single Agent Architectures



- ReAct (Reason + Act)
  주어진 작업에 대해 생각 작성 -> 생각을 바탕으로 행동 -> 결과를 관찰
  => 작업이 완료 될 때 까지 **추론, 관찰, 행동 반복**하여 신뢰성 증가

➔ ReAct 루프에서 벗어나지 못하는 경우 발생 가능
해결방안) 작업 수행 중 인간의 피드백 통합

Single Agent Architectures



- RAISE
  ReAct 방법을 기반으로 하며, 인간의 **단기 및 장기 기억**을 모방하는 **메모리 메커니즘**이 추가된 방법
  단기 정보 : 스크래치패드에 저장
  장기 정보 : 이전의 비슷한 사례들을 데이터셋으로 저장

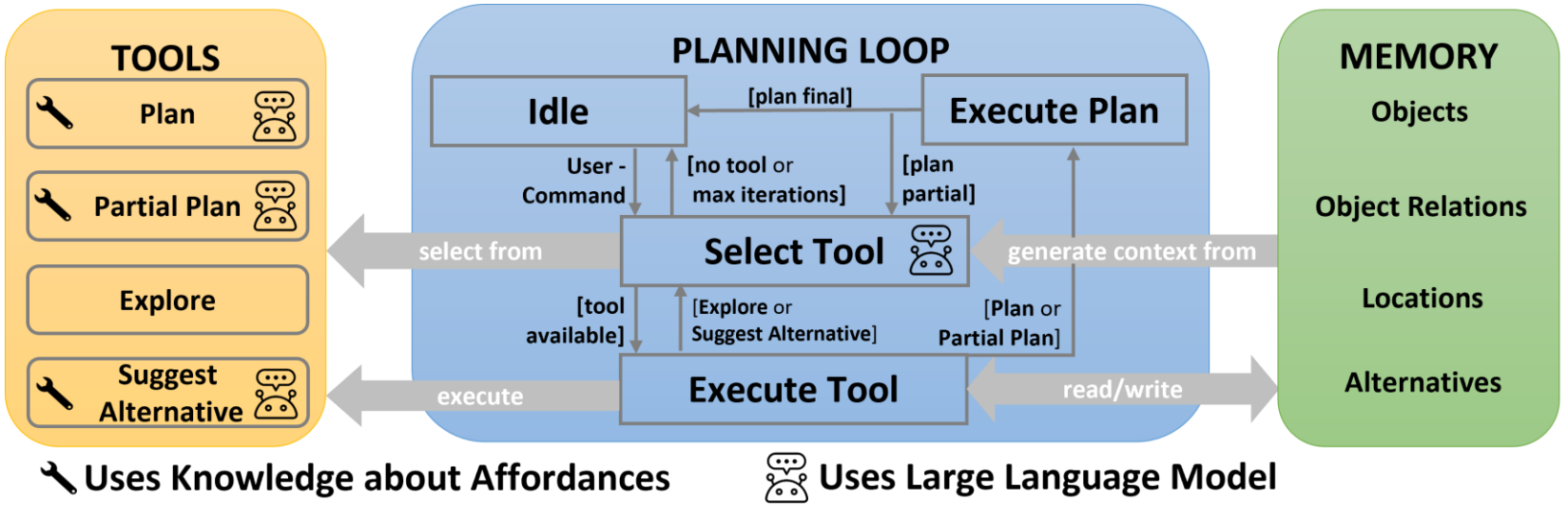
➔ 복잡한 논리 이해 어려움으로 많은 시나리오에서 유용성 제한, 자신의 역할이나 지식에 대해 잘못된 정보 생성
해결방안) 모델을 세밀하게 조정하여 해결

Single Agent Architectures

- Reflexion
  언어적 피드백을 통한 **자기 반성을 활용**하는 단일 에이전트 패턴.
  성공 상태, 현재 진행 경로, 지속적인 기억과 같은 지표를 활용하여 LLM 평가자가 에이전트에게 구체적이고 관련성 있는 피드백을 제공하여 성공률 향상.
  Chain-of-Thought와 ReAct에 비해 허위 생성이 감소.

➔ "Non-optimal local minima solutions“ 취약, 슬라이딩 윈도우 사용하여 장기 기억 용량 제한
해결방안) 다양성, 탐색 및 추론이 중요한 작업에서 성능을 개선할 여지

Single Agent Architectures



- AutoGPT + P (Planning)
  자연어로 로봇을 제어하는 에이전트의 **추론 제한을 해결하는 방법**으로 객체 탐지와 객체 활용도 맵핑을 LLM 기반 계획 시스템과 결합.

  이미지 분석을 통해 객체를 탐지
  계획 도구(Plan Tool) / 부분 계획 도구(Partial Plan Tool) / 대안 제시 도구(Suggest Alternative Tool) / 탐색 도구(Explore Tool) 도구 중 하나를 선택
  목표를 완료하기 위한 전체 계획을 생성

➔ 도구 선택의 정확도 문제와 탐색에서의 비논리적인 결정, 제한된 인간 상호작용

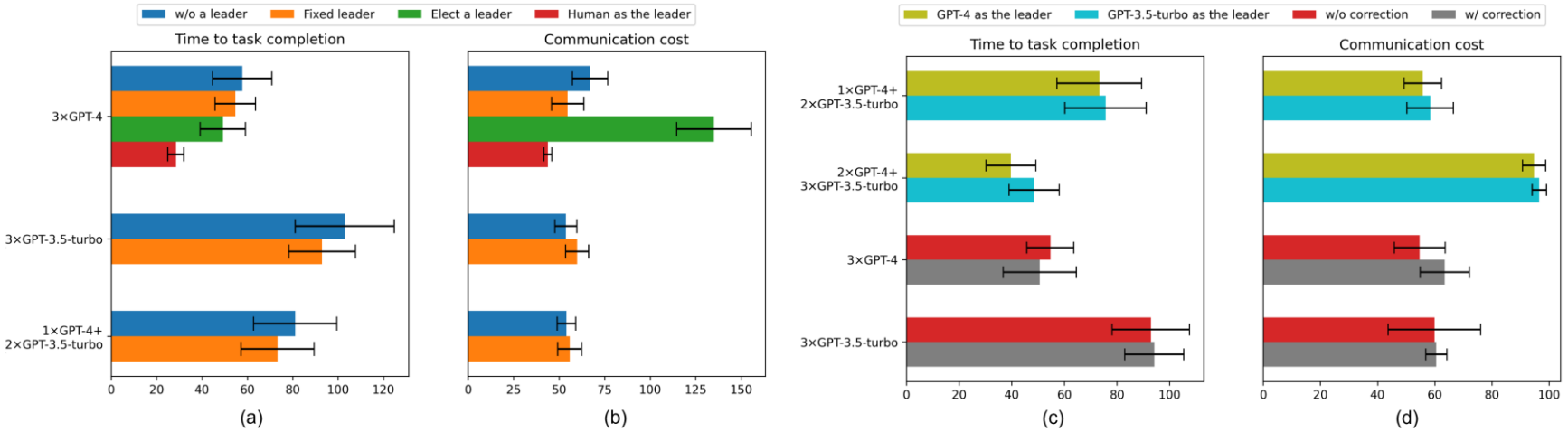
Single Agent Architectures

- LATS (Language Agent Tree Search)
  계획, 행동, 추론을 결합한 단일 에이전트 방식으로, 트리 구조를 사용하여 작업을 수행.

  자기 반성적 추론
  : 행동을 취한 후 환경의 피드백과 언어 모델의 피드백을 통해 추론 오류 확인 후 대안 제시
  효율적인 검색 알고리즘
- : 강력한 검색 알고리즘과 결합된 자기 반성 능력으로 다양한 작업에서 뛰어난 성능

➔ 고급 계산 자원과 긴 처리 시간을 요구, 복잡한 시나리오에 대한 테스트가 부족

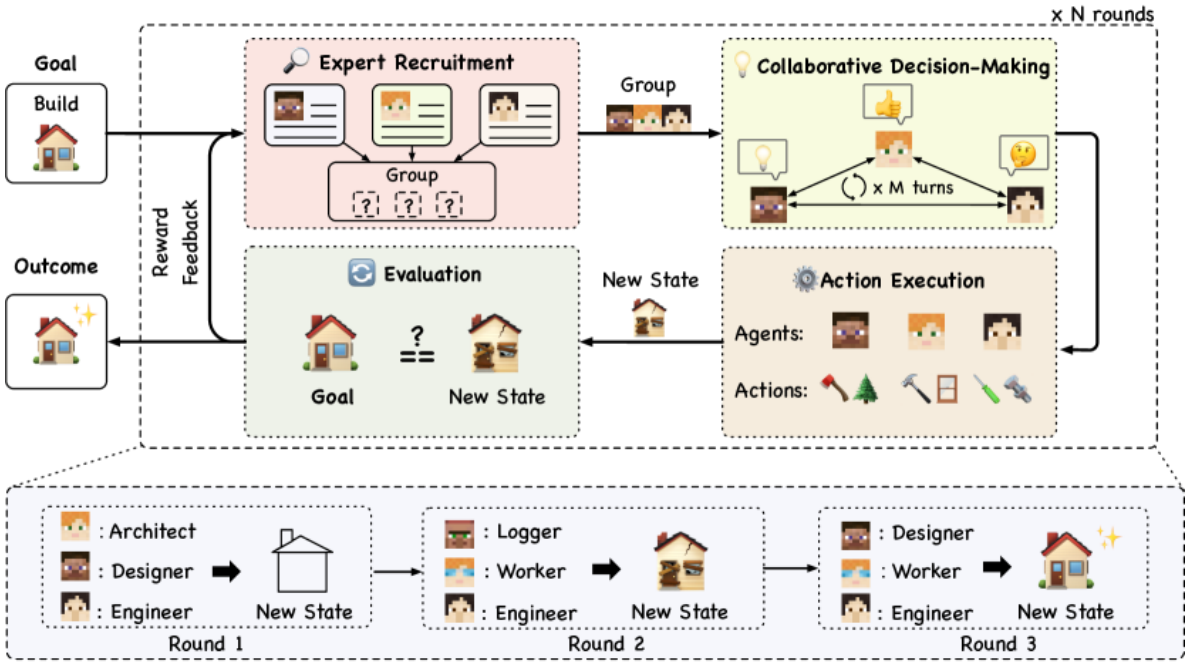
Multi-Agent Architectures



- Embodied LLM Agents Learn to Cooperate in Organized Teams (Guo et al.의 연구)
  리더 에이전트가 에이전트 팀의 전반적인 효율성에 미치는 영향을 보여줌.
  계획 생성, 성능 평가, 피드백 제공, 팀 재구성을 위한 **"비판-반성" 단계를 적용**하는 것의 중요성을 강조.
  리더십과 동적 팀 구조 -> 추론, 계획, 작업 수행을 더 효과적으로 할 수 있도록 개선.

- DyLAN(Dynamic LLM-Agent Network)
  추론 및 코드 생성과 같은 복잡한 작업에 초점을 맞춘 동적 에이전트 구조 생성.
  각 에이전트가 지난 작업에서 얼마나 기여했는지 평가 -> 다음 작업 수행 시 상위 기여자 선택(수평적 성격)
  산술 및 일반 추론 능력을 측정하는 다양한 벤치마크에서 향상된 성과 표출.

Multi-Agent Architectures



- AgentVerse
  그룹 **계획**을 위한 **단계**들이 에이전트의 **추론** 및 **문제 해결 능력**을 어떻게 향상시킬 수 있는지 보여줌.

  모집, 협업적 의사결정, 독립적 행동 실행, 평가
  (recruitment, collaborative decision making, independent action execution, evaluation)
  위 과정은 전체 목표가 달성될 때 까지 반복

Multi-Agent Architectures

- MetaGPT
  에이전트들 간의 생산적이지 않은 대화 문제 해결을 위해,
  에이전트 간 비구조화된 채팅을 공유하는 대신, publish-subscribe(발행-구독) 매커니즘 구현하여 정보 공유

토론과 관찰

에이전트 구현은 문제를 반복적으로 해결하기 위해 **계획 – 실행 – 평가** 과정 따름.

단일 에이전트와 멀티 에이전트 아키텍처 모두 복잡한 목표 실행에서 뛰어난 성과를 보인다고 판단.

두 아키텍처 모두 명확한 피드백, 작업 분해, 반복적인 개선, 역할 정의가 에이전트 성능을 향상시킨다는 사실을 발견.

- 단일 에이전트 vs 멀티 에이전트 아키텍처 선택을 위한 일반적인 조건
- 에이전트와 비동기 작업 실행
- 피드백과 인간의 감독이 에이전트 시스템에 미치는 영향
- 그룹 대화 및 정보 공유의 도전 과제
- 역할 정의 및 동적 팀의 영향

현재 연구의 한계 및 향후 연구 고려사항

- 에이전트 평가의 문제
- 데이터 오염 및 정적 벤치마크의 영향
- 벤치마크의 범위와 전이 가능성
- 실제 세계 적용 가능성
- 에이전트 시스템의 편향성과 공정성

결론

복잡한 목표를 협력하여 수행해야 하는 에이전트 팀을 구성 시
다음 요소 중 하나를 갖춘 에이전트를 배치하는 것이 유리

- 명확한 시스템 프롬프트 정의
- 명확한 리더십과 작업 분담
- 전담된 추론/계획-실행-평가 단계
- 동적 팀 구조
- 인간 또는 에이전트 피드백
- 지능적인 메시지 필터링

=> 단일 에이전트 아키텍처나 이러한 전략이 없는 멀티 에이전트 아키텍처보다 성능이 향상될 가능성이 큼