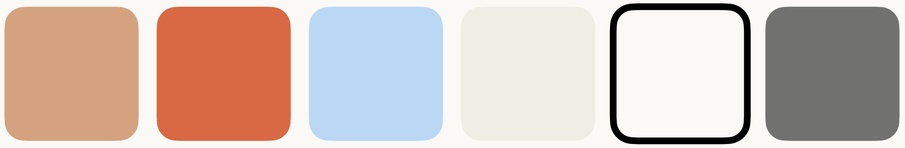
@slowwww.ai



from Anthropic

How we built our multi-agent research system

< The missing manual >

SlowwAI

2025-06-16

@slowwww.ai

Agent 고도화 매뉴얼

사용법

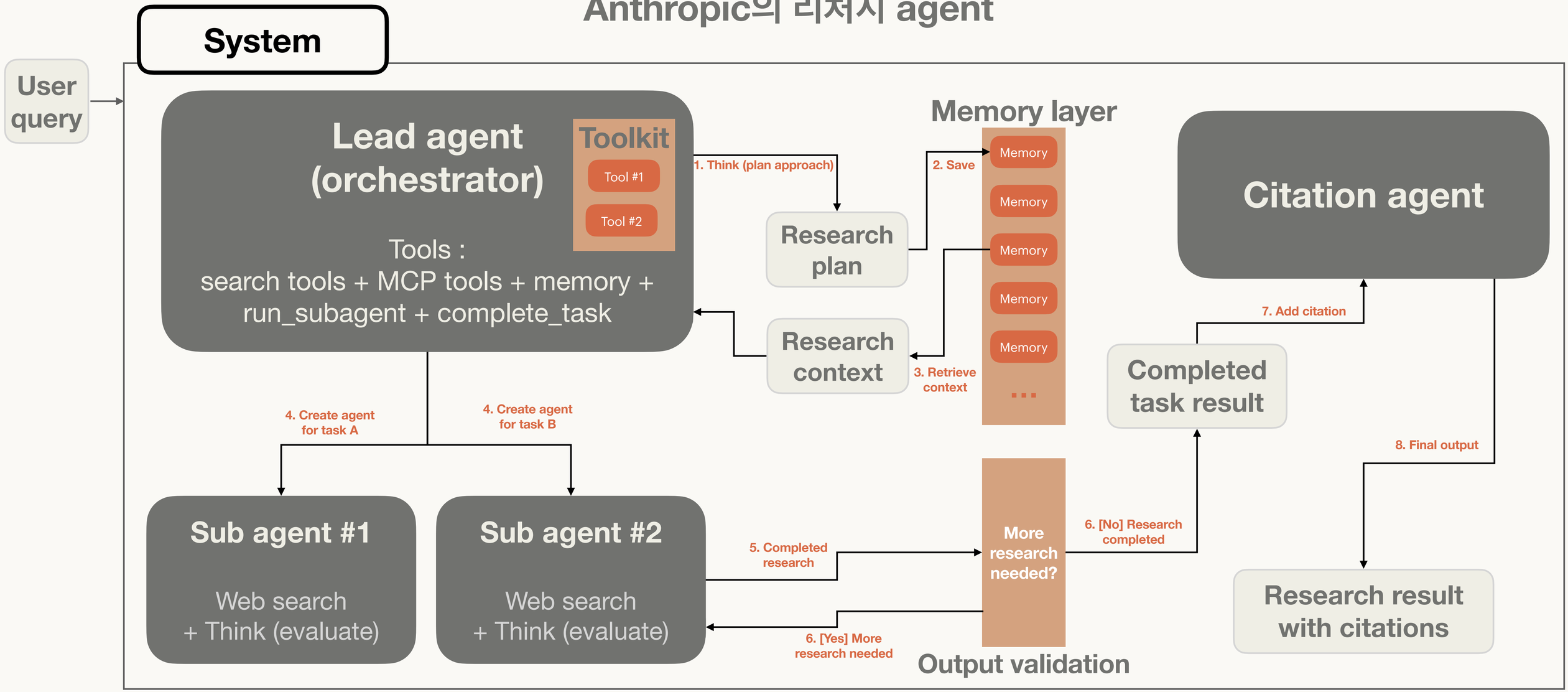
매뉴얼 사용법

1. Anthropic의 agent 구조를 살펴본다
   (System / agent / evaluation)
2. 나의 LLM agent가 부족한 부분이 어디인지 생각해 본다
3. Anthropic은 어떻게 극복했는지 살펴본다
4. Anthropic의 방법을 나의 agent에 적용해본다

전체 아키텍처

@slowwww.ai

Anthropic의 리처시 agent



LLM을 활용한 서비스 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역

에이전트

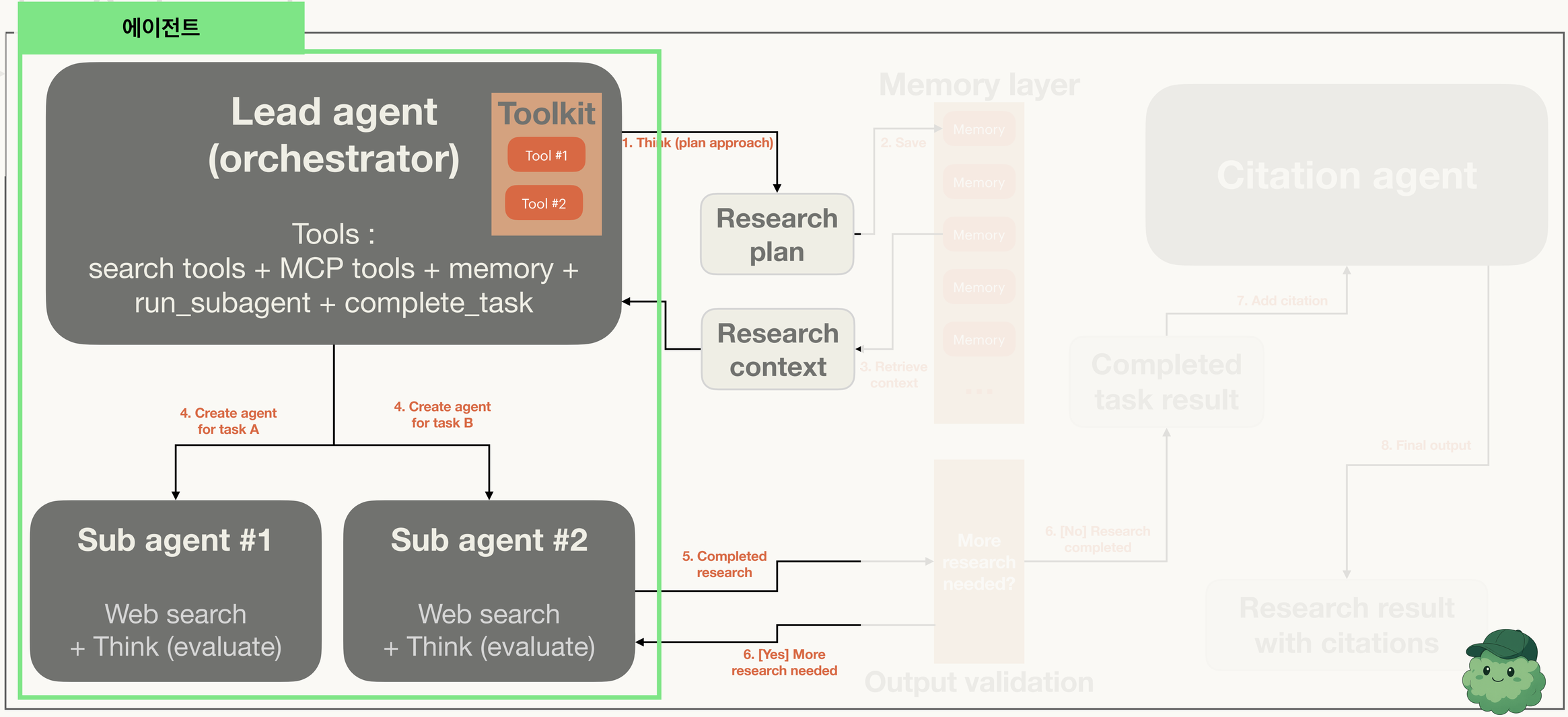
에이전트 평가

시스템 설계
및 배포

@slowwww.ai

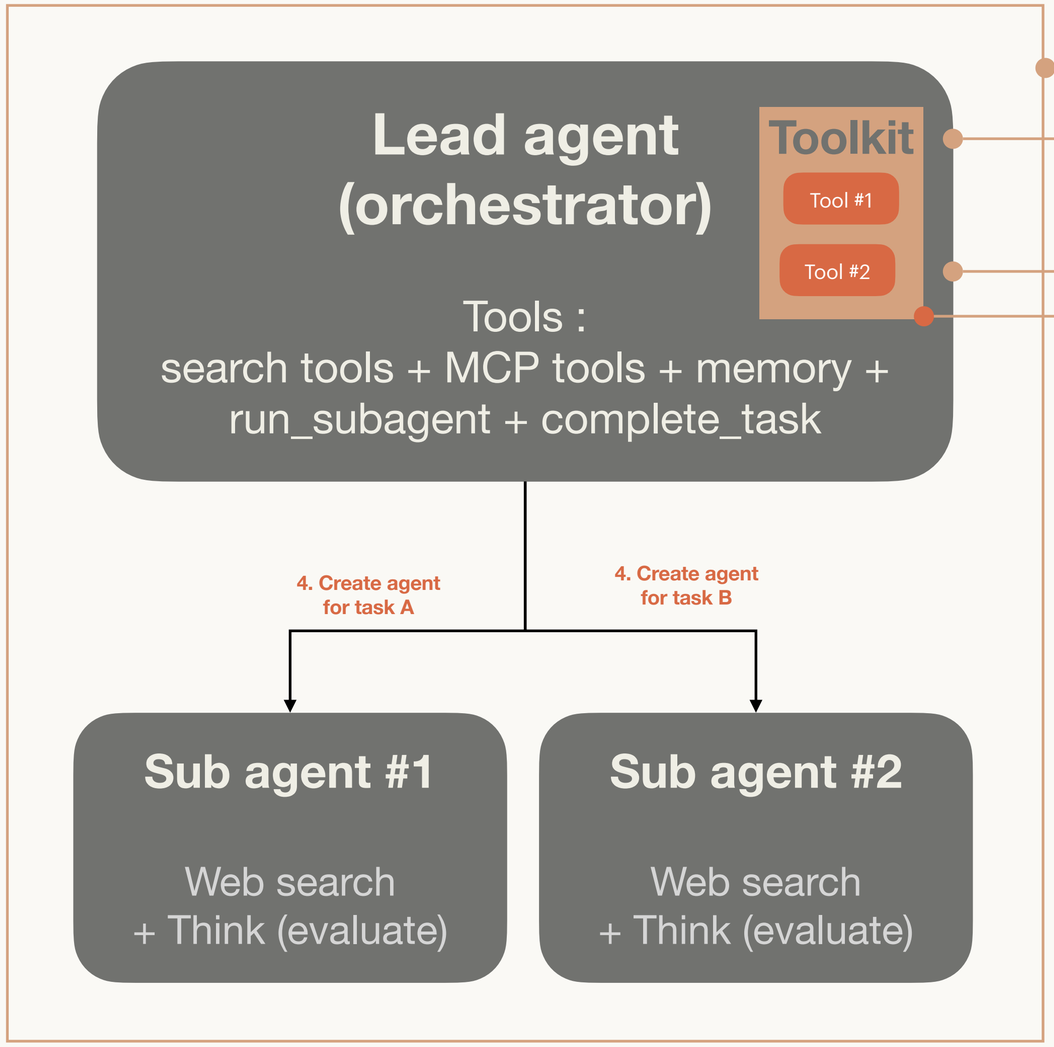
LLM을 활용한 서비스 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역 중 1번째



에이전트 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역 중 1번째



1. 에이전트의 사고 방식을 이해하세요

- 에이전트가 어디에서 실패하는지 파악하는 것이 굉장히 중요합니다.
- 따라서 **input, 중간 과정, 그리고 output 모두 로깅**하여 꼼꼼하게 살펴봐야 합니다

2. Lead agent의 역할이 굉장히 중요합니다

- Lead agent는 팀장처럼 sub agent에게 task를 제공해줍니다.
- 각 sub agent가 “목적”, “결과물 형태”, “활용 가능 툴”, 그리고 “역할”을 제대로 부여하는지 체크해보세요. 말 그대로 R&R을 분명하게 줄 수 있어야 합니다

3. 복잡도에 따라 적응할 수 있어야 합니다

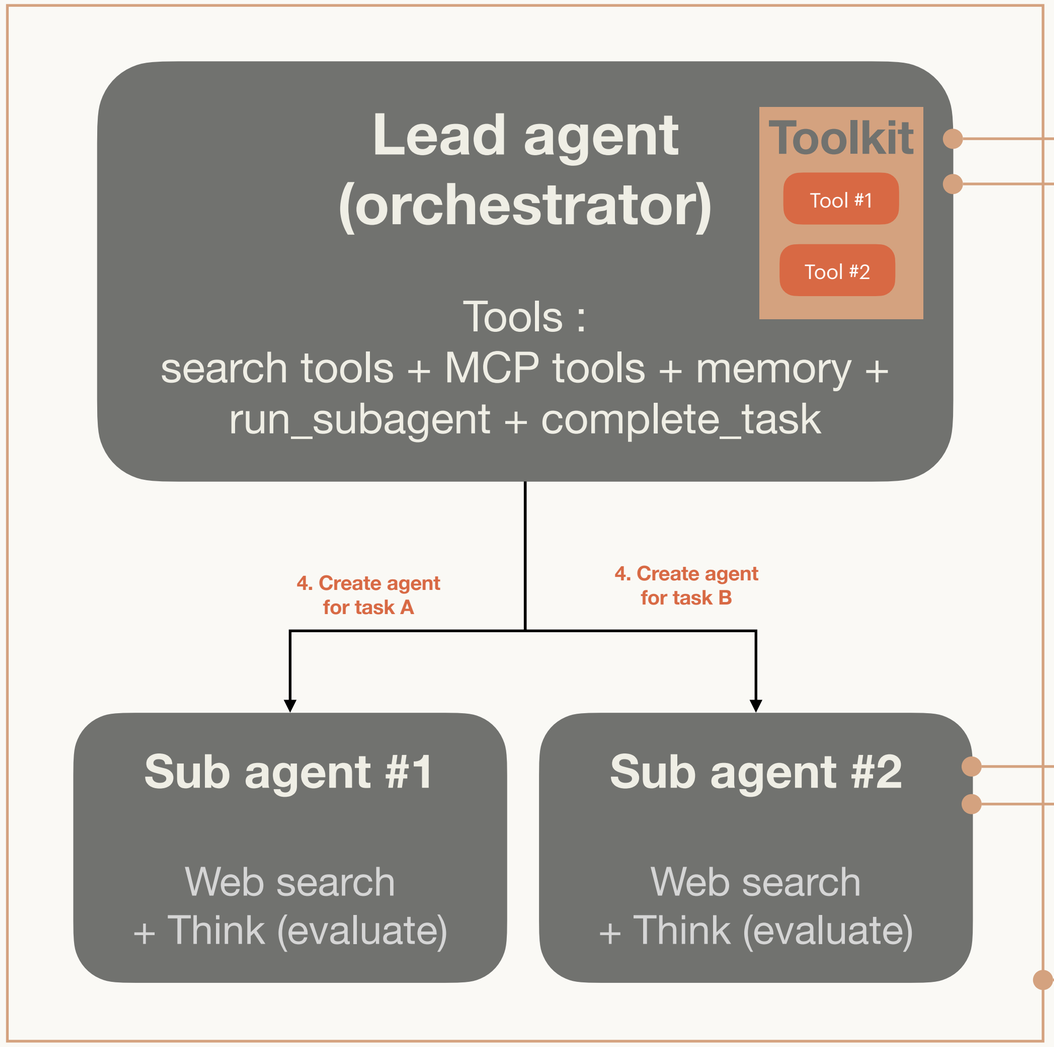
- 간단한 검색 : 3-10개의 도구를 사용할 수 있는 agent 1개를 생성
- 비교 분석 : 5-10개의 도구를 사용할 수 있는 agent 2에서 4개를 생성
- 복잡한 연구 : 명확한 R&R을 가진 agent 10개 이상을 생성

4. 에이전트 스스로 성장하게끔 만드세요

- 에이전트는 도구의 ‘설명’ 부분을 보고 선택 여부를 결정합니다. 그런 ‘설명’이 상황에 맞게끔 적혀있도록 수정하는 에이전트를 만들어 보세요.

에이전트 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역 중 1번째



5. [프롬프트] 넓게 시작해서 좁혀가도록

- 에이전트에게 긴 쿼리를 던지면, 전체를 처리하지 못하고 짧은 대답만 내놓는 경우가 많습니다.
- 따라서 처음에는 추상적인 넓은 답변을 내놓고, 점진적으로 구체적인 task를 만들어 나가도로고 프롬프트를 조정해보세요.

6. [프롬프트] LLM의 생각 프로세스를 도와주세요

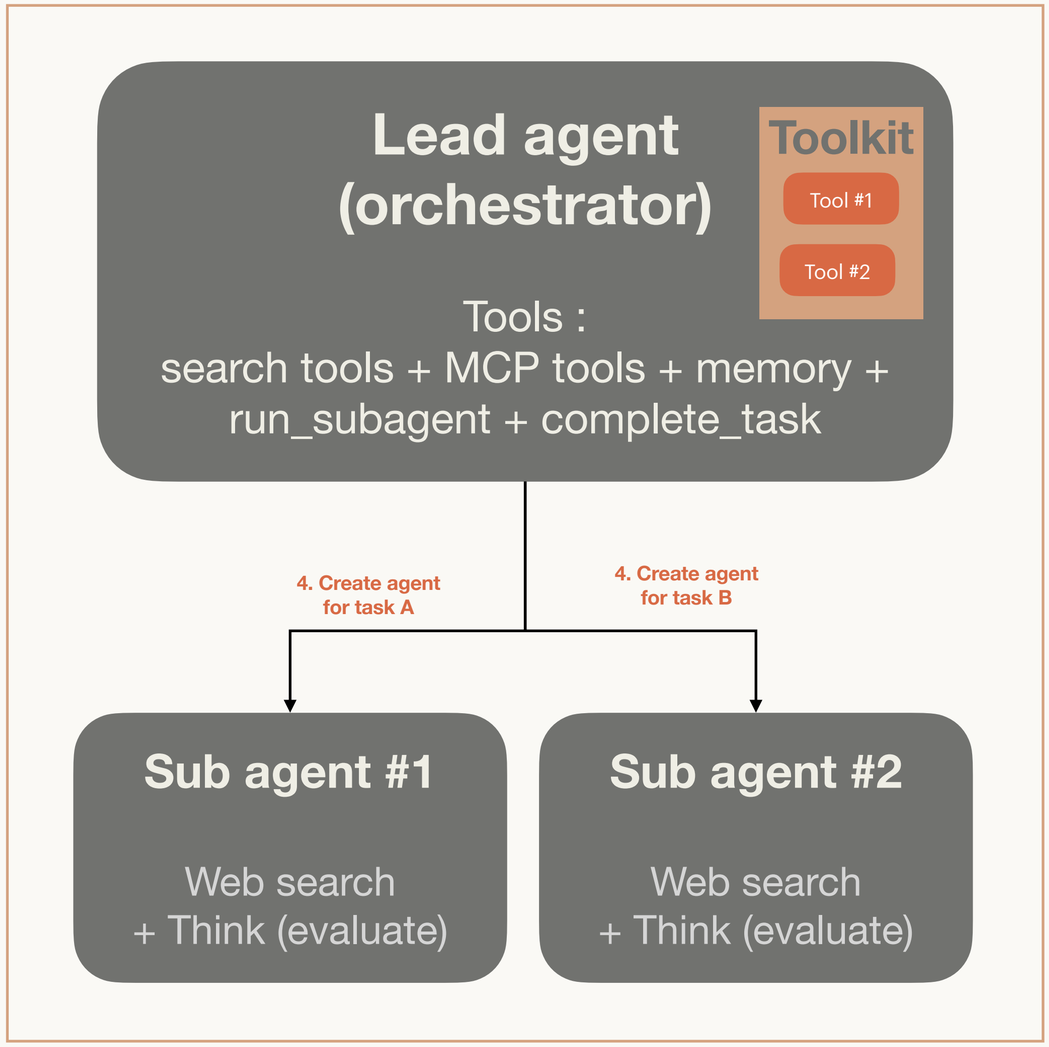
- Claude의 “extended thinking” 모드를 활성화 하세요. 설명에 따르고, 논리적으로 생각하기 때문에 효율성이 올라갑니다
- 다른 모델이라면, 생각하는 프로세스를 직접 프롬프트에 입력하여 가이드 해주세요.

7. 도구 사용은 병렬적으로 실행하도록 하세요

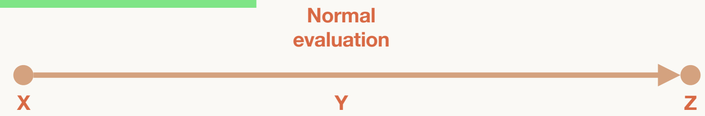
- Lead agent는 3-5개의 도구를 동시에 실행하여 결과를 받고
- Subagent들은 3개 이상의 도구를 동시에 실행하도록 만들어 보세요
- 각 도구들로 부터 결과들을 동시에 받을 수 있기 때문에, 전체적으로 agent가 답변을 내놓는 속도가 훨씬 빨라 집니다

에이전트 평가 방식의 고도화

퍼포먼스를 올릴 수 있는 세 가지 영역 중 2번째

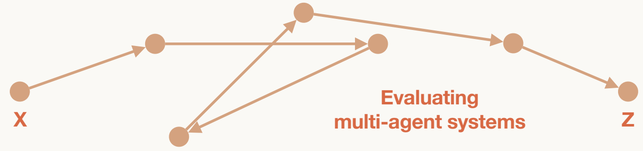


Evaluation



멀티 에이전트 시스템을 평가하는 방법은 기존의 시스템들을 평가하는 것과 다릅니다.

기존의 AI 시스템은 X에서 시작하여 Y를 거치고, Z로 가는 등, 매번 유사한 방법으로 최종 아웃풋을 생성해낸다는 가정 하세 evaluation을 진행합니다. 하지만 멀티 에이전트는 굉장히 다이나믹 하기 때문에 이런 전통적인 평가 방법으로 평가할 수 없습니다.



동일한 데이터를 갖고 시작하더라도, 에이전트가 목표를 달성하는 방법은 천차만별입니다. 어떤 에이전트는 세 개의 자료만 찾아보고 끝내지만, 다른 에이전트는 열 개를 뒤지거나 전혀 다른 도구로 같은 답을 얻기도 합니다.

애초에 '정답 경로'를 항상 알 수 없으니, 미리 정해 둔 "올바른 절차"를 그대로 밟았는지만 확인해서는 안됩니다. 따라서 결과 정답 여부는 물론, 동시에 그 과정이 합리적이었는지까지 살필 수 있는 유연한 평가 방식이 필요합니다.

에이전트 평가 방식의 고도화

퍼포먼스를 올릴 수 있는 세 가지 영역 중 2번째

1. 에이전트 구축과 동시에 평가도 시작하세요

- 프로젝트 초반에는 작은 수정에도 굉장히 많은 부분을 개선할 수 있습니다.
- 따라서 20개의 쿼리 정도만이라도 활용하여 지속적으로 현재 에이전트를 평가하고 개선하세요.
- 그렇지 않으면 점점 복잡도가 올라가 평가하기가 어려워집니다.

2. 잘만 만들면 LLM을 활용해서 평가를 할 수 있습니다

- 사람이 모든 케이스를 평가할 수는 없습니다. 다음과 같이 LLM을 활용하여 평가해보세요.
- 하나의 프롬프트로 LLM을 한 번 호출해 0.0–1.0 점수와 합격·불합격 판정을 동시에 추출해보세요. 평가 지표는 사실 정확성, 인용 정확성, 출처 품질, 도구 효율성을 기준으로 평가 됩니다.

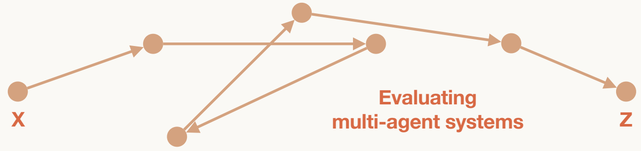
3. 그래도 사람이 직접 평가해야 하는 부분도 있습니다

- 전체적인 퍼포먼스는 LLM을 활용하여 측정할 수 있겠지만, 사람에 의해서만 찾을 수 있는 특정 edge case들이 있습니다.
- 특이한 쿼리에 대한 답변이나, 답변 방식, 또는 미묘하게 편향된 답변 등이 있을 겁니다.
- 이런 일반화된 패턴을 찾아 수정할 수 있어야 합니다.

Evaluation

멀티 에이전트 시스템을 평가하는 방법은 기존의 시스템들을 평가하는 것과 다릅니다.

기존의 AI 시스템은 X에서 시작하여 Y를 거치고, Z로 가는 등, 매번 유사한 방법으로 최종 아웃풋을 생성해낸다는 가정 하세 evaluation을 진행합니다. 하지만 멀티 에이전트는 굉장히 다이나믹 하기 때문에 이런 전통적인 평가 방법으로 평가할 수 없습니다.



동일한 데이터를 갖고 시작하더라도, 에이전트가 목표를 달성하는 방법은 천차만별입니다. 어떤 에이전트는 세 개의 자료만 찾아보고 끝내지만, 다른 에이전트는 열 개를 뒤지거나 전혀 다른 도구로 같은 답을 얻기도 합니다.

애초에 '정답 경로'를 항상 알 수 없으니, 미리 정해 둔 "올바른 절차"를 그대로 밟았는지만 확인해서는 안됩니다. 따라서 결과 정답 여부는 물론, 동시에 그 과정이 합리적이었는지까지 살필 수 있는 유연한 평가 방식이 필요합니다.

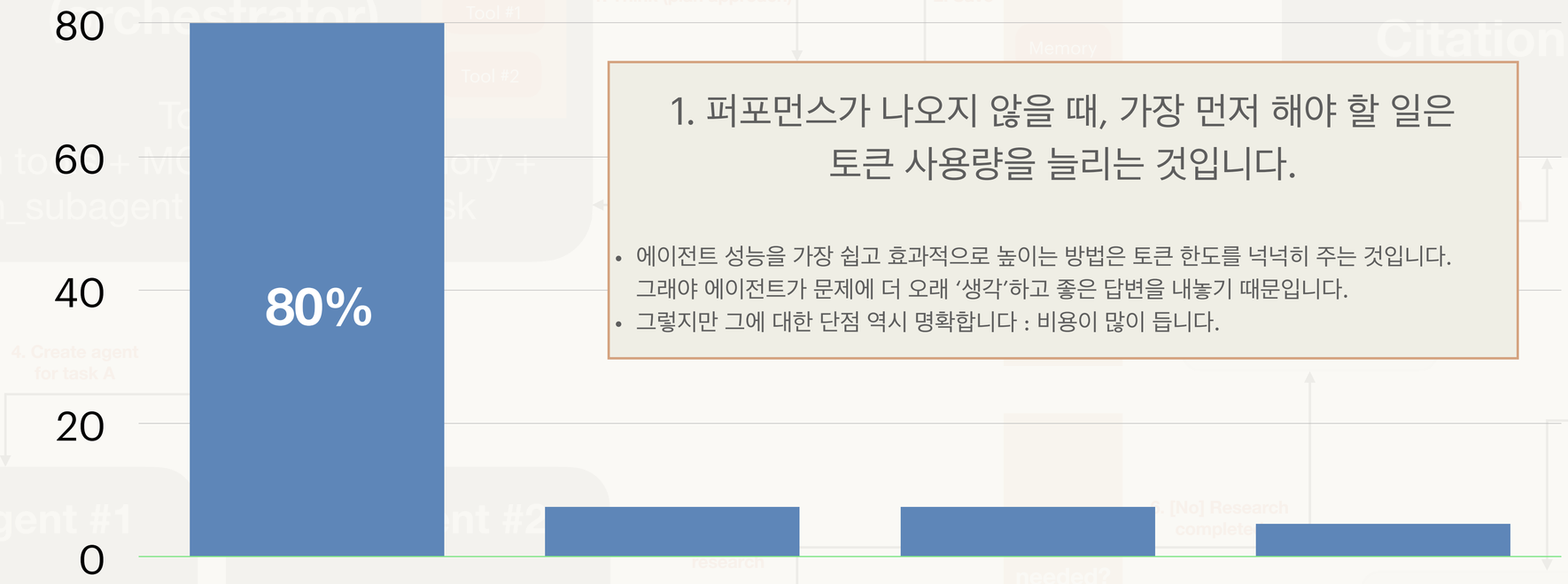
시스템 설계 및 배포의 고도화 방법

Production level

System

퍼포먼스를 올릴 수 있는 세 가지 영역 중 3번째

전체 퍼포먼스를 결정짓는 요소

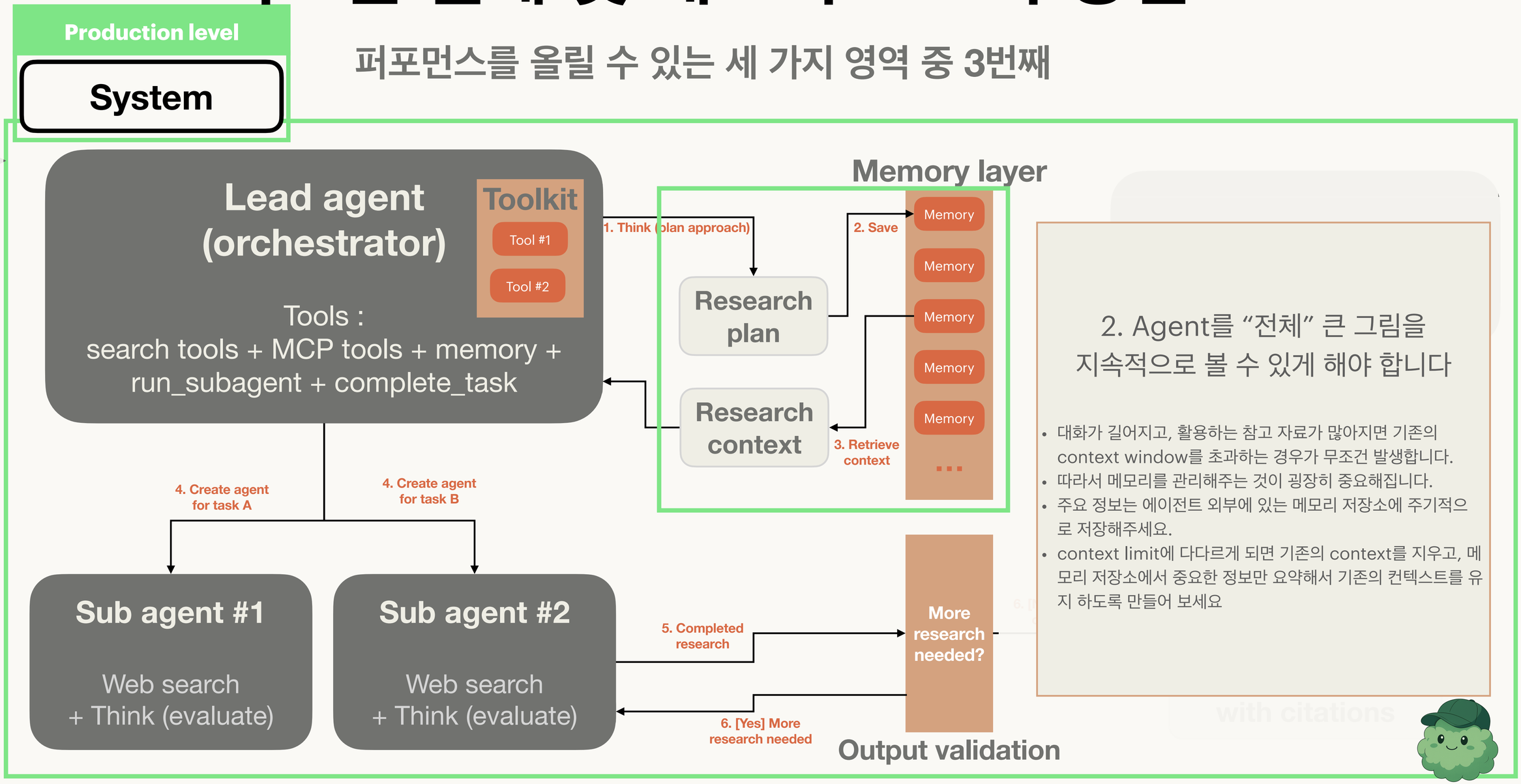


1. 퍼포먼스가 나오지 않을 때, 가장 먼저 해야 할 일은 토큰 사용량을 늘리는 것입니다.

- 에이전트 성능을 가장 쉽고 효과적으로 높이는 방법은 토큰 한도를 넉넉히 주는 것입니다. 그래야 에이전트가 문제에 더 오래 '생각'하고 좋은 답변을 내놓기 때문입니다.
- 그렇지만 그에 대한 단점 역시 명확합니다 : 비용이 많이 듭니다.

시스템 설계 및 배포의 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역 중 3번째

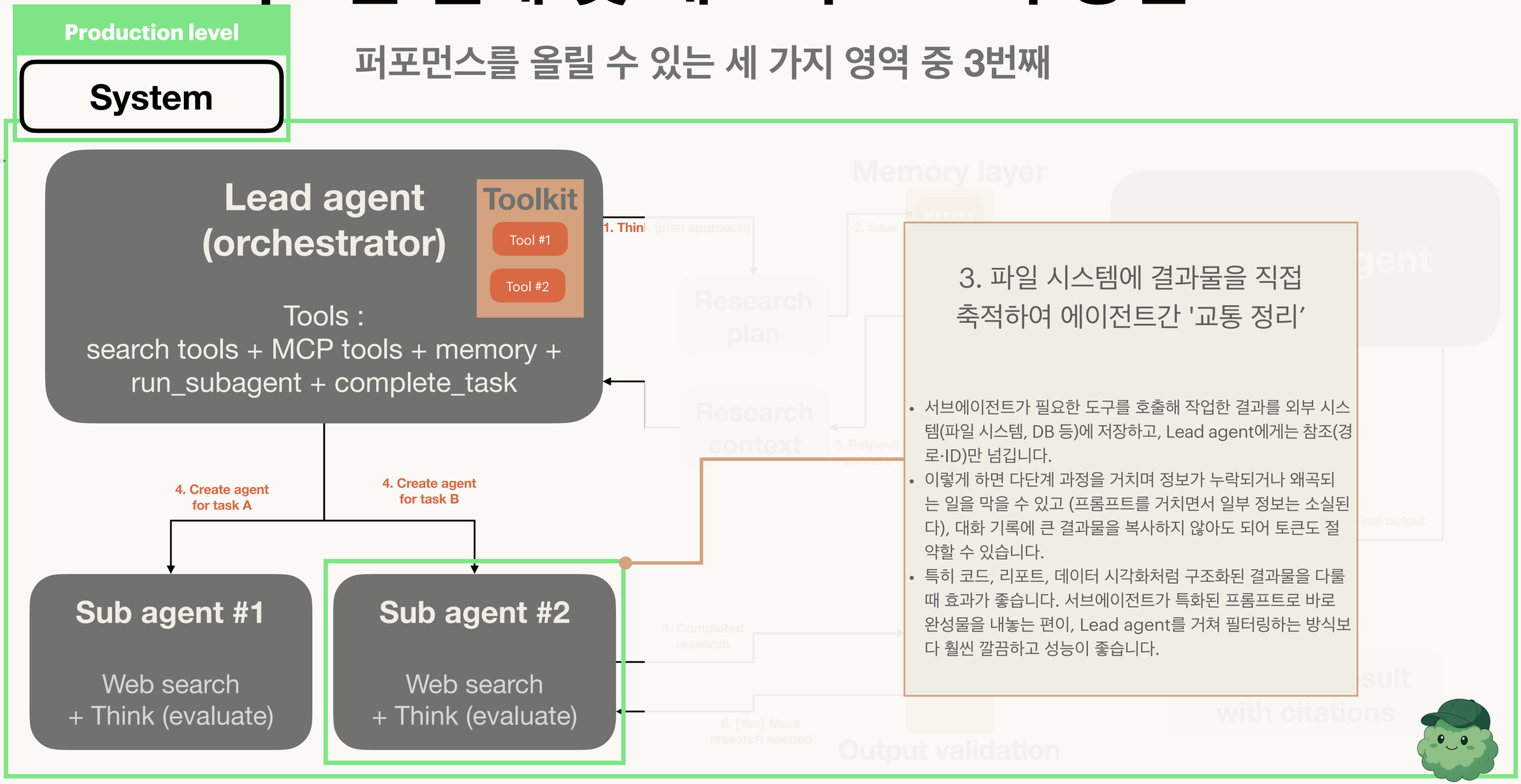


2. Agent를 “전체” 큰 그림을 지속적으로 볼 수 있게 해야 합니다

- 대화가 길어지고, 활용하는 참고 자료가 많아지면 기존의 context window를 초과하는 경우가 무조건 발생합니다.
- 따라서 메모리를 관리해주는 것이 굉장히 중요해집니다.
- 주요 정보는 에이전트 외부에 있는 메모리 저장소에 주기적으로 저장해주세요.
- context limit에 다다르게 되면 기존의 context를 지우고, 메모리 저장소에서 중요한 정보만 요약해서 기존의 컨텍스트를 유지 하도록 만들어 보세요

시스템 설계 및 배포의 고도화 방법

퍼포먼스를 올릴 수 있는 세 가지 영역 중 3번째



3. 파일 시스템에 결과물을 직접 축적하여 에이전트간 '교통 정리'

- 서브에이전트가 필요한 도구를 호출해 작업한 결과를 외부 시스템(파일 시스템, DB 등)에 저장하고, Lead agent에게는 참조(경로·ID)만 넘깁니다.
- 이렇게 하면 다단계 과정을 거치며 정보가 누락되거나 왜곡되는 일을 막을 수 있고 (프롬프트를 거치면서 일부 정보는 소실된다), 대화 기록에 큰 결과물을 복사하지 않아도 되어 토큰도 절약할 수 있습니다.
- 특히 코드, 리포트, 데이터 시각화처럼 구조화된 결과물을 다룰 때 효과가 좋습니다. 서브에이전트가 특화된 프롬프트로 바로 완성물을 내놓는 편이, Lead agent를 거쳐 필터링하는 방식보다 훨씬 깔끔하고 성능이 좋습니다.

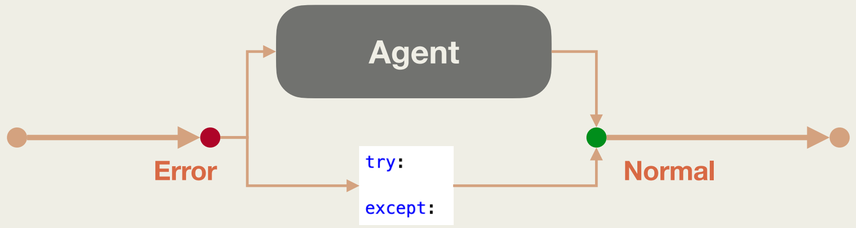
시스템 설계 및 배포의 고도화 방법

Production level

System

퍼포먼스를 올릴 수 있는 세 가지 영역 중 3번째

4. 각 에이전트는 이전의 에이전트의 결과물을 바톤터치 받습니다. 자연스럽게 오류들도 바톤터치 받습니다.



- 에이전트는 맥락이 중요합니다. 그렇기 때문에 오류가 나면 그 즉시 수정해줄 필요가 있습니다.
- 그런데 오류가 나올 때마다 처음부터 다시 돌리면 비용이 많이 들어가고, 사용자도 오래 기다려야 합니다.
- 그렇다면 에러가 난 지점에서 바로 이어서 실행할 수 있는 복구 체계를 추가해보세요.
- 엔트로픽은 어떤 툴이 실패했음을 에이전트에게 알려 주면 해당 에이전트는 스스로 전략을 바꿔 진행하도록 만드는 방식으로 문제를 유연하게 처리하도록 설계 했습니다.
- 기본적인 try-except로 예외를 잡아 재시도하거나 fall-back 로직을 태우는 것도 좋은 방법 입니다.

5. 에이전트의 행동 패턴을 파악 해보세요

- 에이전트의 행동 패턴을 파악하면, 어떤 부위를 고도화 해야 하는지 구체적으로 알 수 있습니다. 그러기 위해서는 logging은 필수 중의 필수 입니다.
- 단순한 메트릭이나 로그를 넘어서, 에이전트의 의사결정 흐름과 상호작용 구조까지 모니터링해야 구체적인 고도화 포인트를 찾을 수 있습니다.
- 예상치 못한 결과를 파악하거나, 특정 문제의 원인을 탐색하고, 자주 일어나는 문제점을 해결하는데 많은 도움을 줄 겁니다.

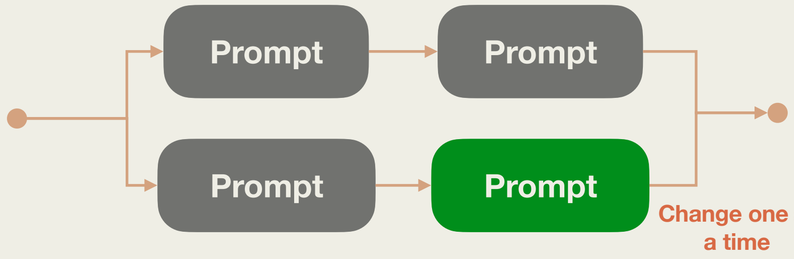
시스템 설계 및 배포의 고도화 방법

Production level

System

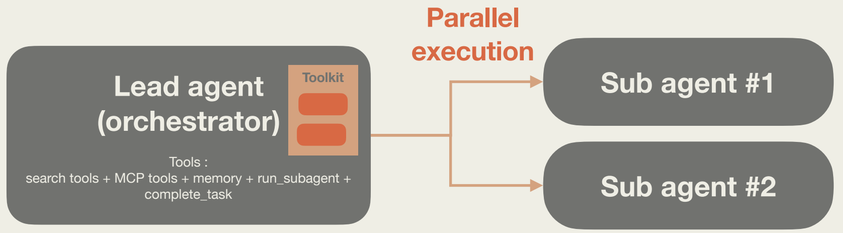
퍼포먼스를 올릴 수 있는 세 가지 영역 중 3번째

6. 배포는 점진적으로 & 고립시켜 진행해야 합니다



- Agent를 업데이트 할 때에는 항상 조심해야 합니다. 하나의 에이전트가 다른 에이전트를 실행시키고, 결국 엄청난 나비효과가 일어날 수 있기 때문입니다.
- 따라서 잘 작동하는 agent들은 그대로 두되, 고도화가 필요한 부분만 고립시켜 업데이트 하는 것이 중요합니다.
- 이렇게 고립을 유지하면서 진행을 해야 나비 효과가 적어지고, 최종 결과에 대한 컨트롤도 가져갈 수 있습니다.

7. 비동기 방식으로 실행을 하세요 (병렬 실행)



- 동기식 실행 방식은 전체 flow를 조율하기 쉬운 장점은 있지만 정보 흐름이 굉장히 느립니다.
- 비동기식으로 돌리면 에이전트들이 동시에 작업하고, 필요할 때 새 서브에이전트를 바로 생성해 속도를 극대화할 수 있습니다. 물론 비동기 환경에서는 결과 조율, 상태 일관성, 오류 전파를 맞추는 일이 더 까다롭긴 합니다.
- 그렇지만 모델이 점점 더 길고 복잡한 연구 과제를 다룰 수 있게 되면서, 그 결과물을 만들기 위한 시간도 자연스럽게 증가하게 됩니다. 실행 시간도 결국 비용이며, 사용자들에게 결과물을 빠르게 전달하는 것도 굉장히 중요합니다.