RAG와 AI Agent의 이해

다양한 LLM Agent의 구조

Previous Summary

LLM의 능력을 증강시키는 (3+1)개의 요소

- Retrieval, Tool, Memory
- Reflection: 다단 추론을 위한 능력

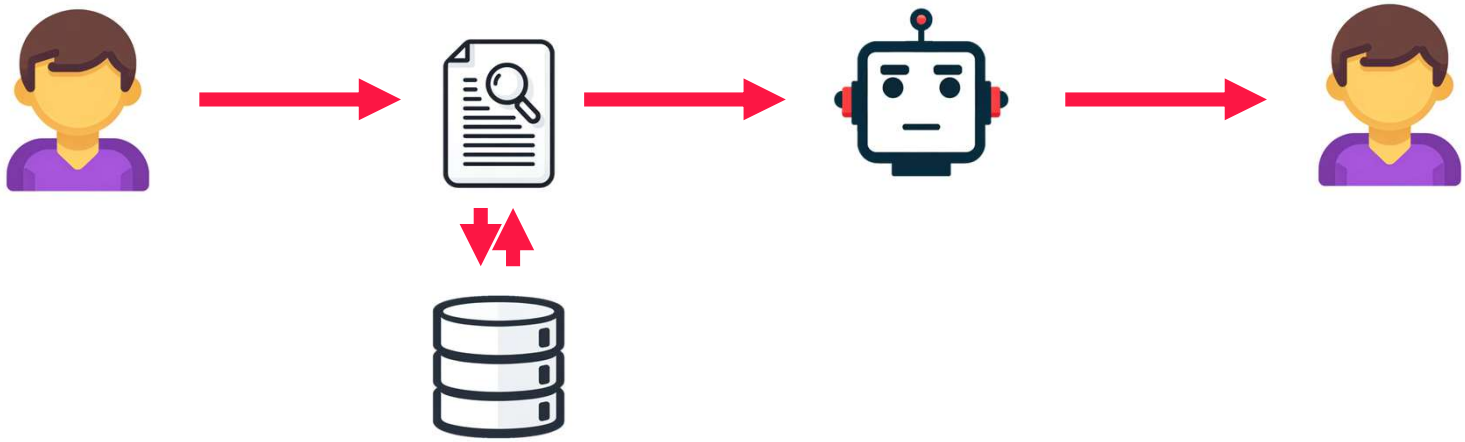
Agent & Workflow → Agentic Work

- 다양한 Agent와 Agentic Work에는 무엇이 있을까?

Retrieval Augmented Generation (RAG)

검색 + 증강 + 생성

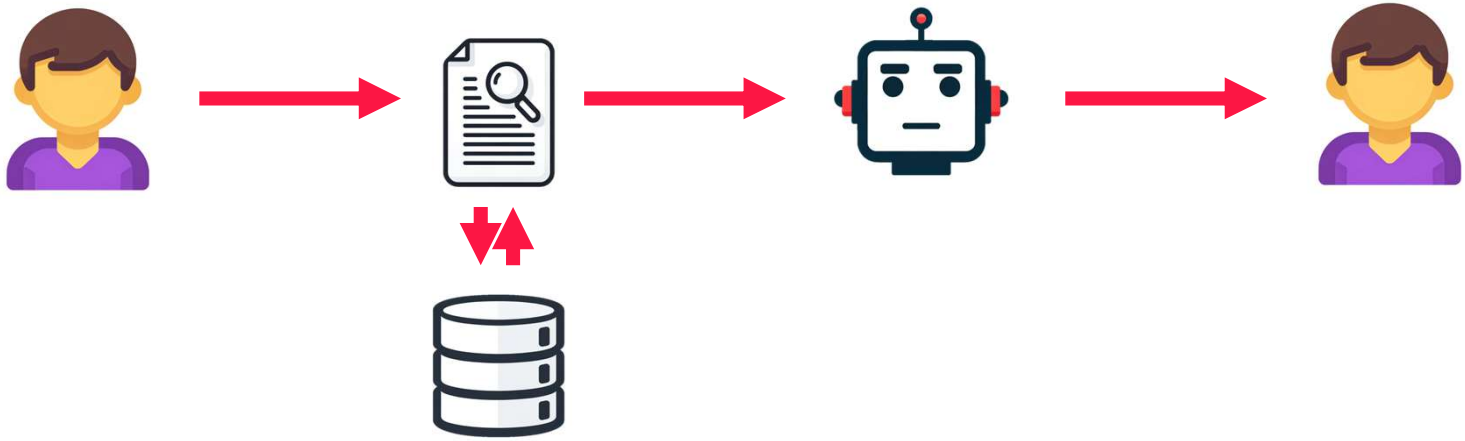
- 사용자의 질문에 대해, 관련 정보를 검색한 뒤 프롬프트에 포함하여 LLM에 전달
- 정보를 바탕으로 답변한 LLM은 할루시네이션을 줄일 수 있음



RAG의 동작을 증강시키는 Agent

RAG의 다양한 시나리오 고려하기

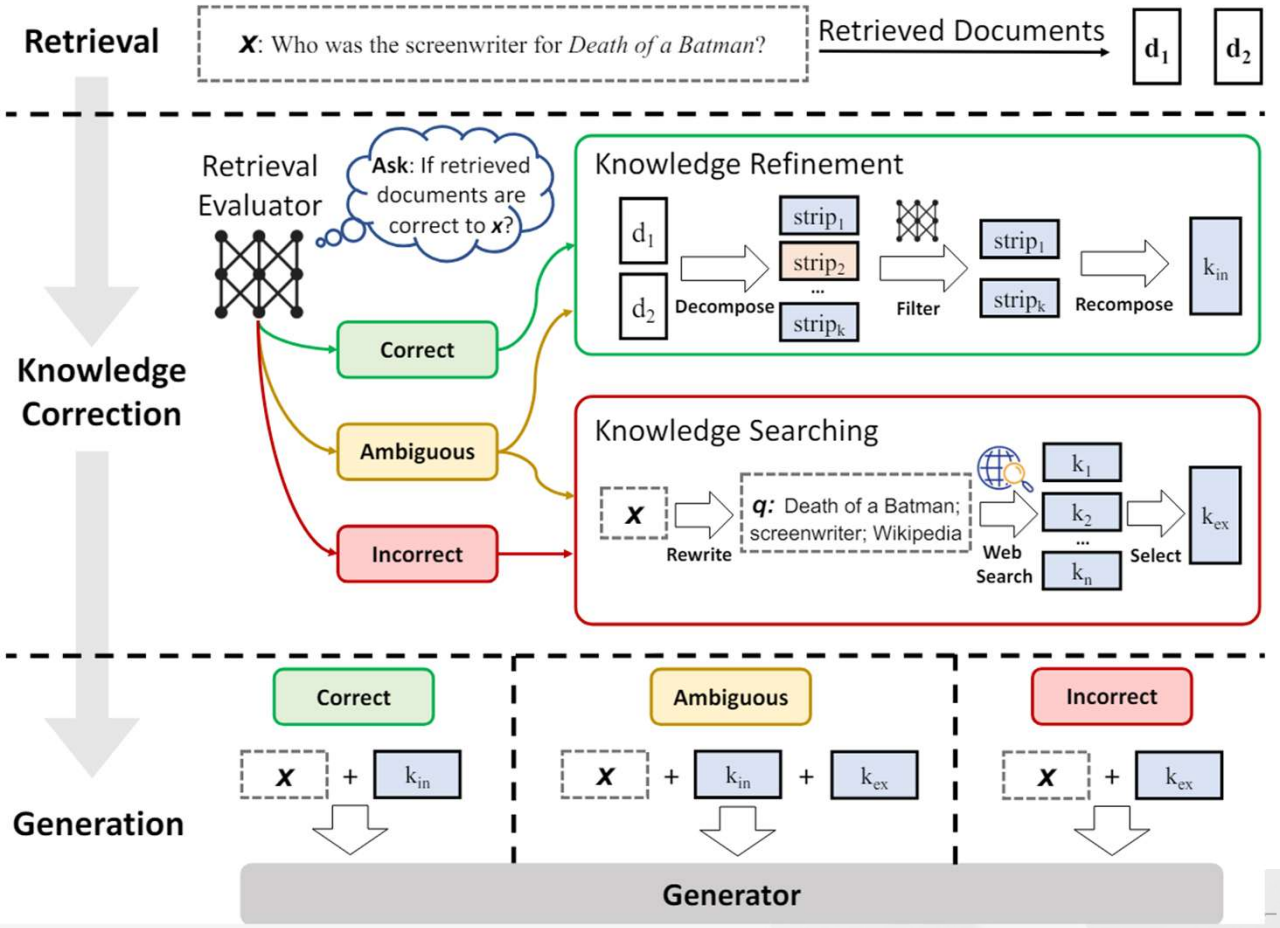
- 만약 검색 결과가 질문 답변에 도움이 안 되는 경우에는?
- 처음부터 RAG가 필요하지 않은 질문이라면 어떻게 해야 할까?



Agentic RAG의 예시: Corrective RAG

RAG에 웹 검색 모듈을 추가

- 검색 결과가 정확하지 않으면 웹 검색

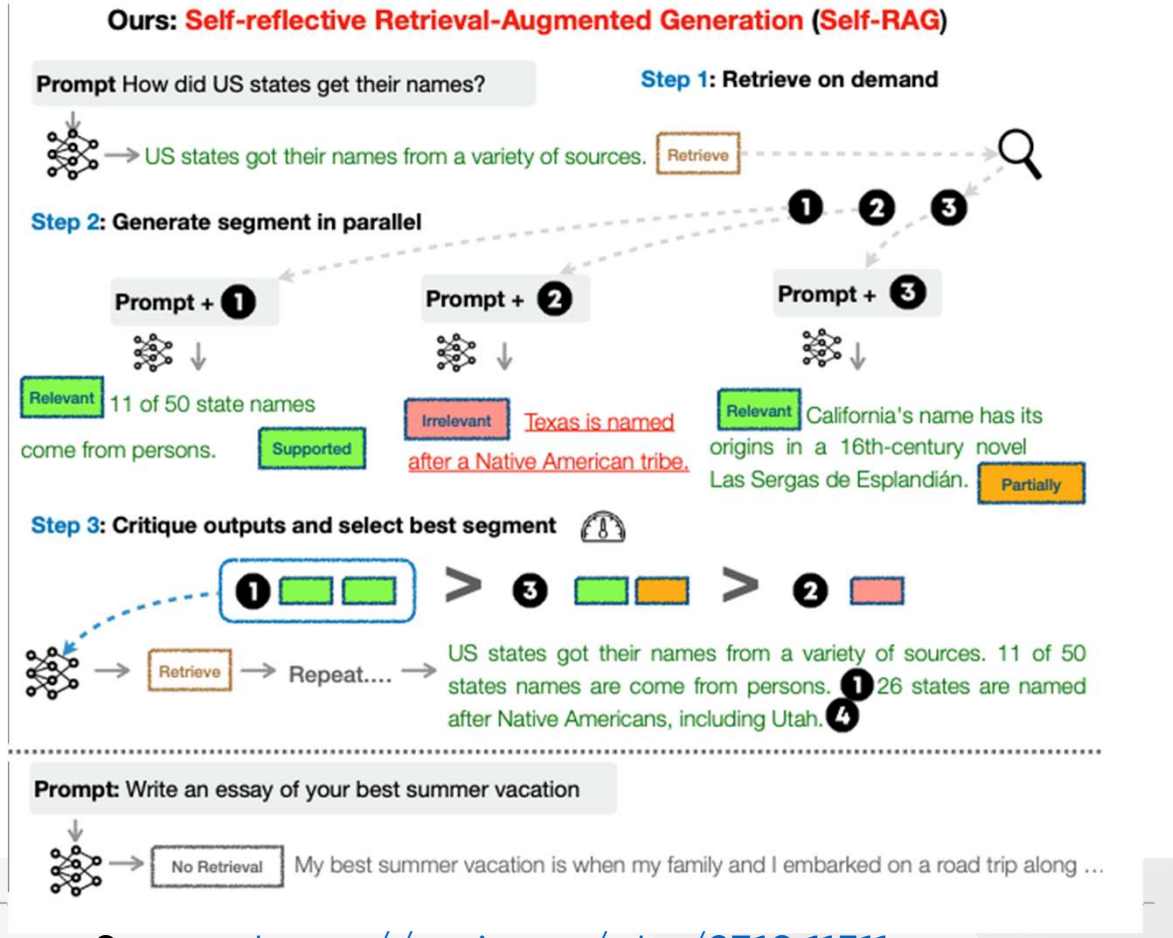


Source: https://arxiv.org/abs/2401.15884

Agentic RAG의 예시: Self-RAG

RAG의 플로우 다각화

- RAG가 필요한 질문인지를 사전에 분류
- 검색 결과를 검증한 뒤 최종 답변 생성



Source: https://arxiv.org/abs/2310.11511

Agentic RAG의 시사점

Application의 다양한 시나리오를 고려하려면

- 사용자 입력/ 데이터베이스/ 캐시 등의 모듈과 LLM의 효과적인 연결 필요

- LLM의 높은 Cost를 고려한 효율적인 플로우 구성

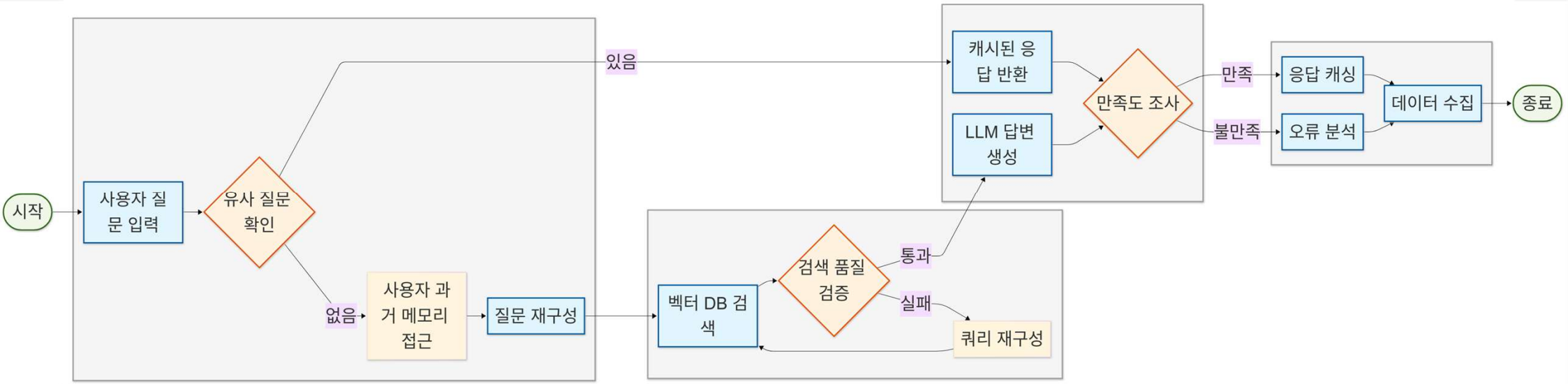
- LLM의 높은 언어 이해 능력을 바탕으로 동적 판단을 내리는 구조 구성
  → LLM은 텍스트 형식의 입력을 받으므로, 프롬프트로의 데이터 변환 필요

Production Level의 RAG Agent

Caching, Memory, Monitoring, Evaluation

- 기존 질문 중 유사한 질문이 있는지 확인하여 LLM 호출 줄이기
- 사용자별 실행 결과를 저장하고, 추후에 활용하기
- 답변 만족도를 조사해 기록하고 개선하기
- RAG 이외의, 다른 작업을 하나의 서버에서 연결하여 사용하기

Production Level의 RAG Agent



Memory와 Human-in-the-loop

복잡한 플로우에서 필요한 능력

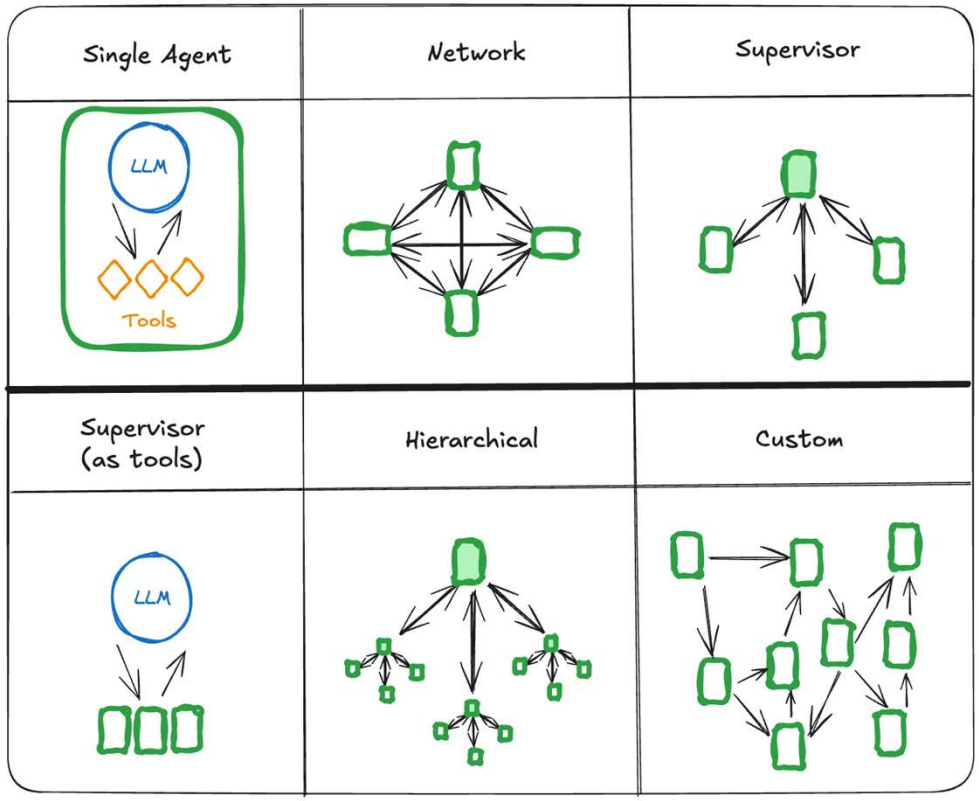
- 과거 내용을 바탕으로 재개하거나, 필요한 정보를 바탕으로 전처리
- 에이전트의 작업에서 중요한 부분이나 수정이 필요한 부분을 인간이 개입

- LangGraph는 이런 측면에서 매우 효율적인 구성을 지원

멀티 에이전트 구조

에이전트 여러 개의 상호 작용

- 1) Supervisor / Worker
  → 여러 에이전트를 병렬로 실행하고 중앙 제어
- 2) Network 구조
  → 정해진 루트보다 그때그때 구성이 필요할 때
- 3) Hierarchical 구조
  → 하나의 Supervisor로 구성하기 복잡할 때



Source: https://langchain-ai.github.io/langgraph/concepts/img/multi_agent/architectures.png

Summary) 언제 어떤 에이전트 구조를 써야 할까?

다양한 예시 학습을 통한 인사이트 필요

- 모든 단일 기능을 독립 요소로 구현하면
  과도하게 복잡한 구조/ 토큰의 과다 소모가 발생할 수 있음
- 큰 기능 단위의 모듈을 구성하고, 이를 연결하는 방법을 구상하기
- 멀티 에이전트/ 복잡한 그래프 구조를 바로 접근하기보다는
  Requirements를 먼저 생각하고 개발하는 소프트웨어 공학적 구성이 동일하게 요구됨