04

LangGarph

RAG의 한계

문서 내 질문에 대한 답변이 존재하지 않는 경우

부족한 정보를 Web 검색하여 문서에 추가하는 로직을 추가

만약 검색결과에 잘못된 정보가 포함되거나 혹은 검색 결과에 없다면?

잘못된 검색결과가 결국 Hallucination으로 이어진다면

RAG의 한계

🤔 검색이 제대로 나올 때까지 반복하여 검색 해볼까?

Hallucination을 방지하는 LLM을 추가해야 하나?

기존 전통적인 RAG의 문제점

1. 문서 업로드

Document Loader

사전에 정의된

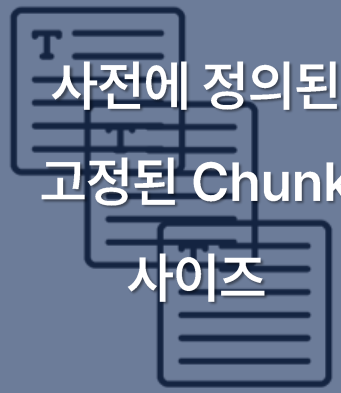
데이터 소싱 자원

(PDF, DB 등)

PyPDFLoader를 활용한 문서 가져오기

2. 문서 분할

Text Splitter



사전에 정의된

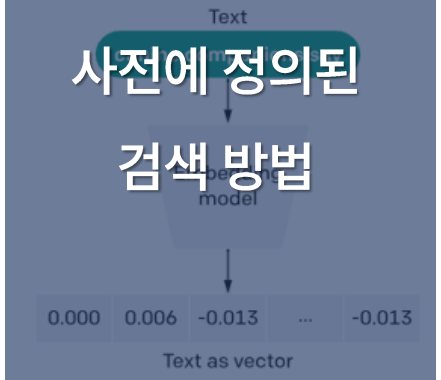
고정된 Chunk

사이즈

PDF 문서를 여러 문서로 분할

3. 문서 임베딩

Embed to Vectorstore



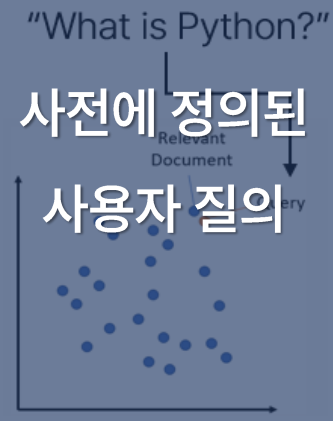
사전에 정의된

검색 방법

LLM이 이해할 수 있도록 문서 수치화

4. 임베딩 검색

VectorStore Retriever



사전에 정의된

사용자 질의

질문과 연관성이 높은 문서 추출

5. 답변 생성

QA Chain

답변 결과에 대한

문서와의 관련성

Prompt 1

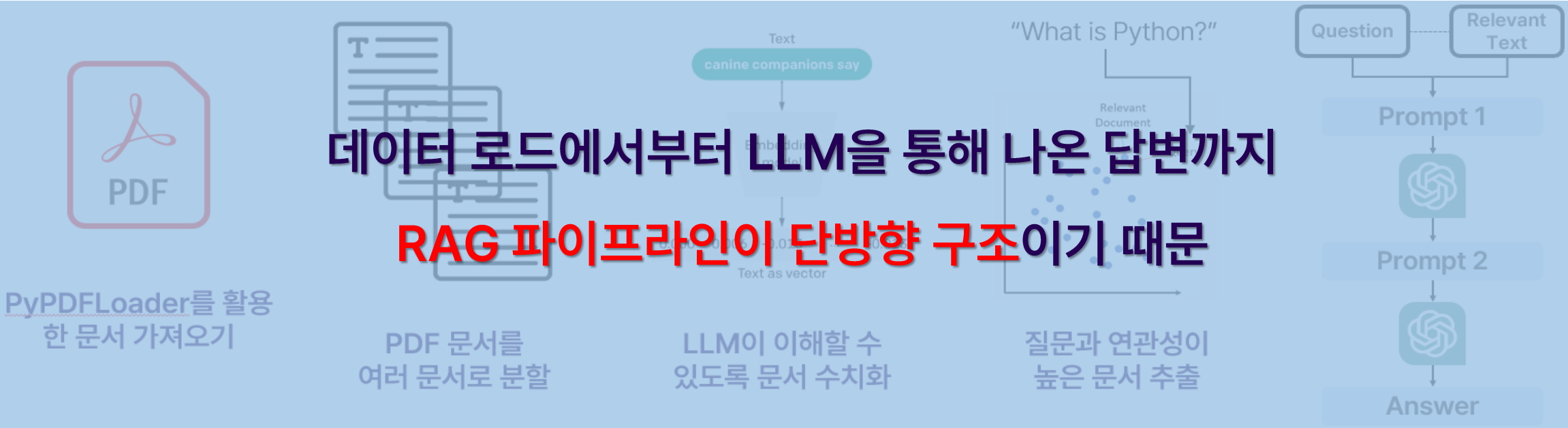
Prompt 2

신뢰하기 어려운

LLM or Agent

Answer

기존 전통적인 RAG의 문제점

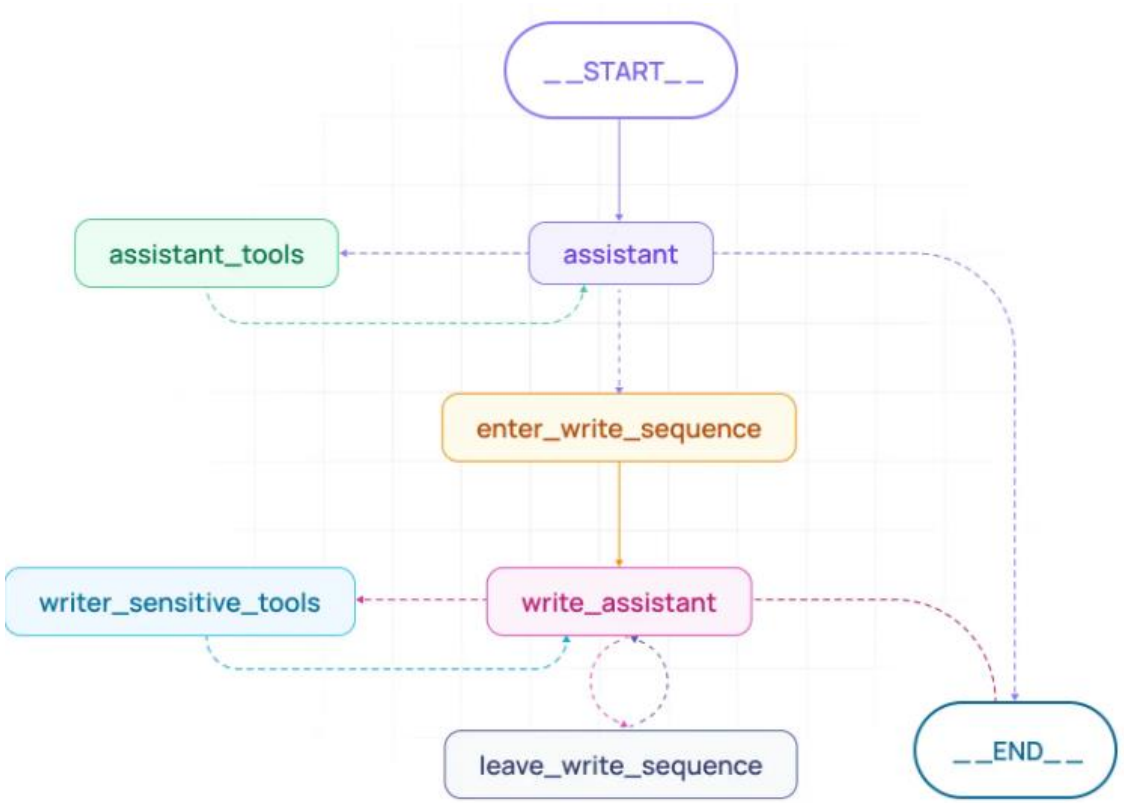


데이터 로드에서부터 LLM을 통해 나온 답변까지

RAG 파이프라인이 단방향 구조이기 때문

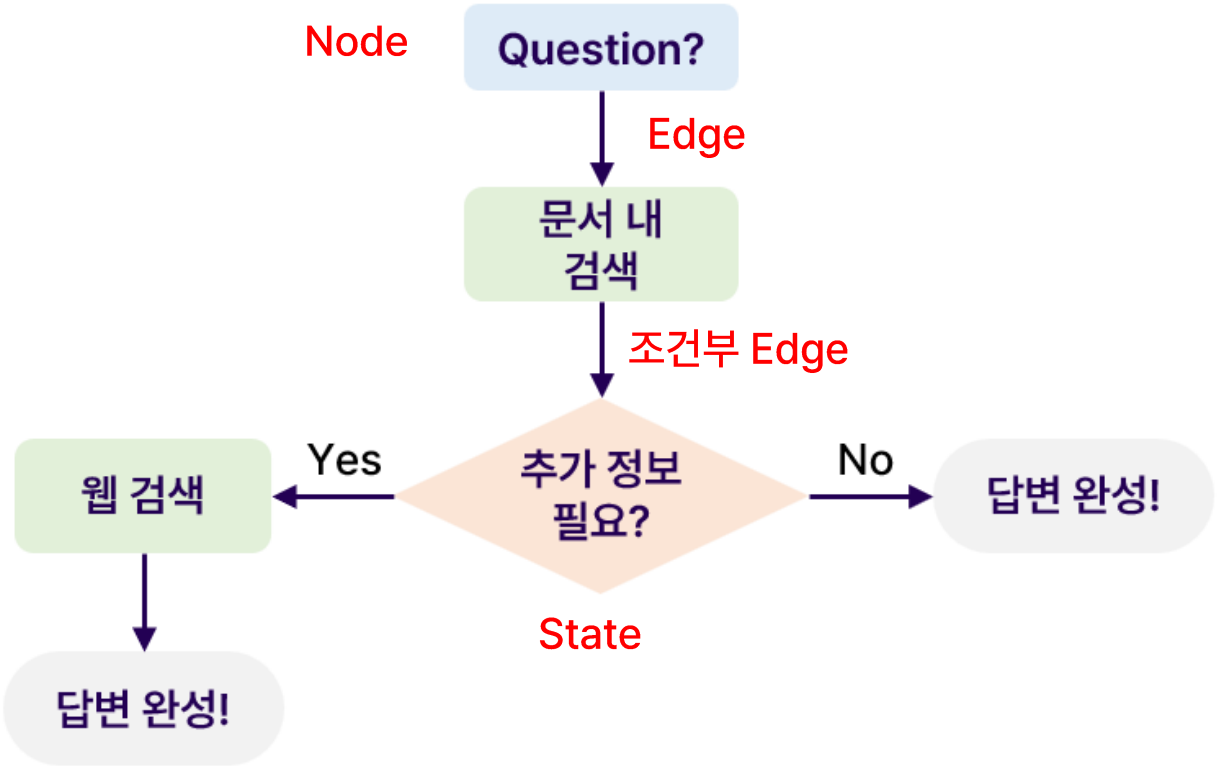
LangGraph

> LangGraph은
>
> 다양한 색(state)으로 표현된 점(node)이
>
> 방향(sign)을 갖은 선(edge)과 만나
>
> 구성된 다이어그램(graph)이다.



source: https://www.langchain.com/langgraph

LangGraph

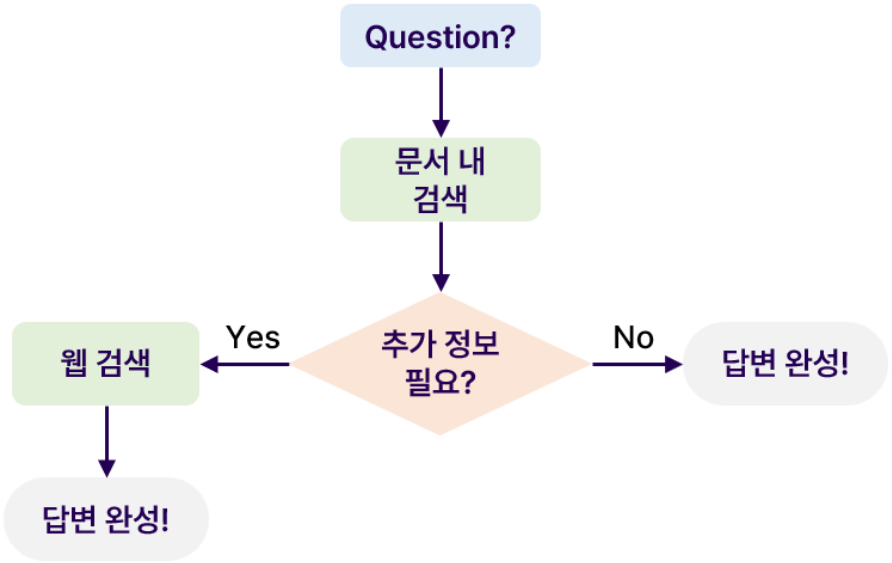


- **Node** 수행하고자 하는 작업 내용
- **Edge** 노드와 노드 사이의 연결
- **조건부 Edge** 를 통해 분기 처리
- **State** 현재의 상태 값을 저장 및 전달하는데 활용

➡️ RAG 파이프라인을 유연하게 설계 가능

source: https://www.langchain.com/langgraph

LLM 을 활용한 워크플로우에 순환(Cycle) 연산 기능을 추가하여 손 쉽게 흐름을 제어



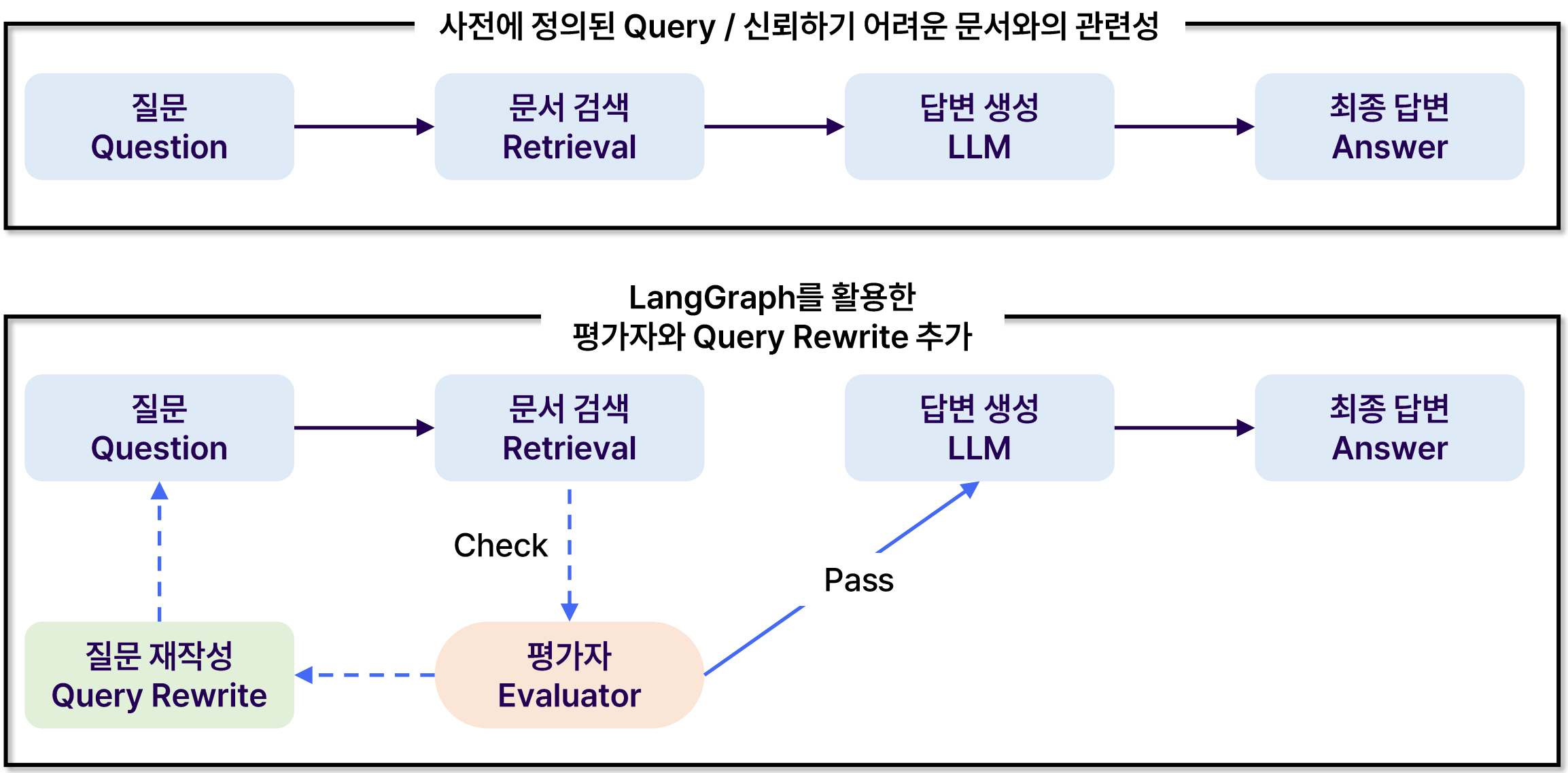
RAG 파이프라인의 세부 단계별 흐름제어가 가능

Conditional Edge: 조건부 (if, elif, else 와 같은..) 흐름 제어

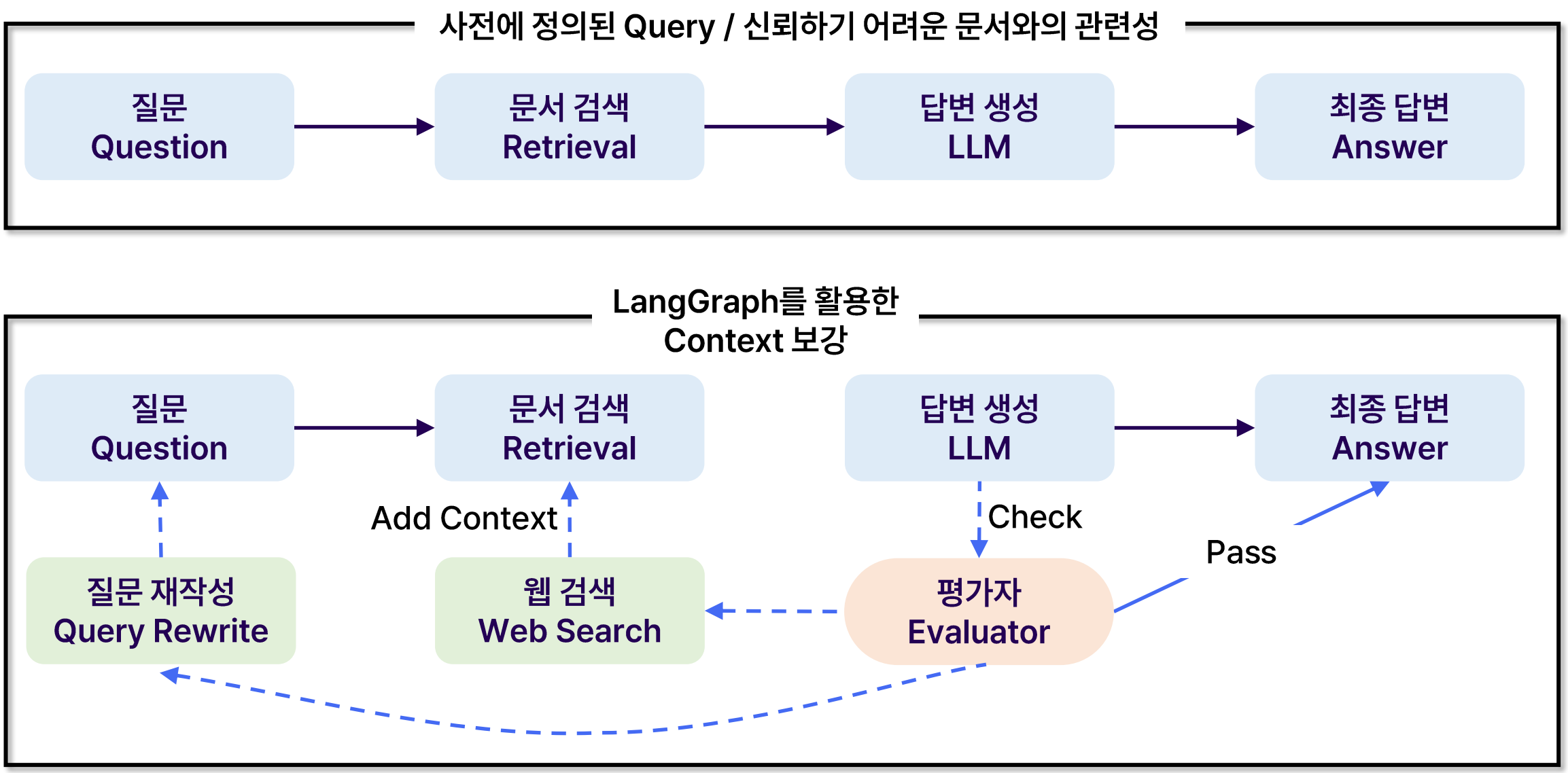
Human-in-the-loop: 필요시 중간 개입하여 다음 단계를 결정

Checkpointer: 과거 실행 과정에 대한 “수정” & “리플레이” 기능

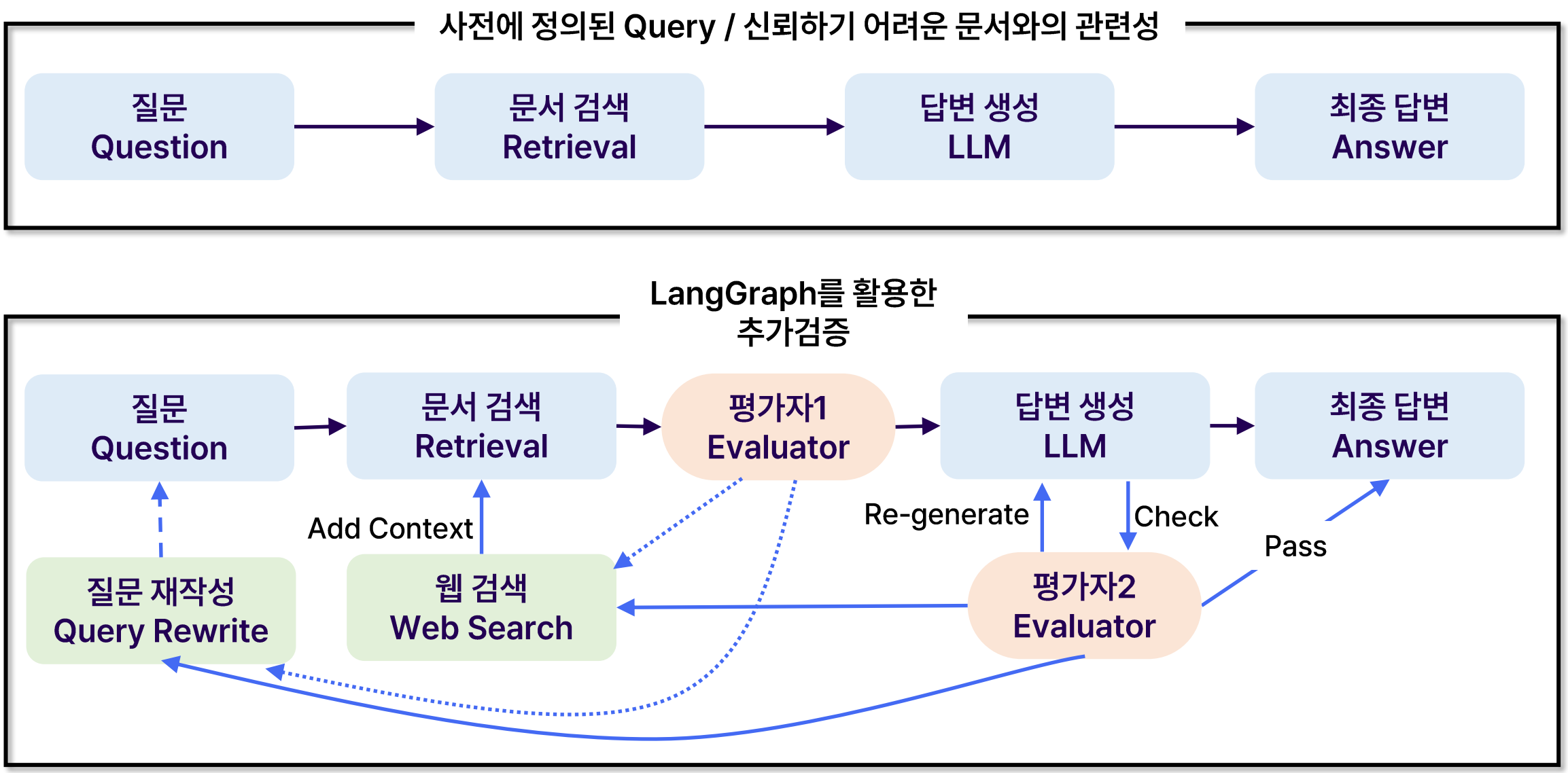
LangGraph를 활용하면?



LangGraph를 활용하면?



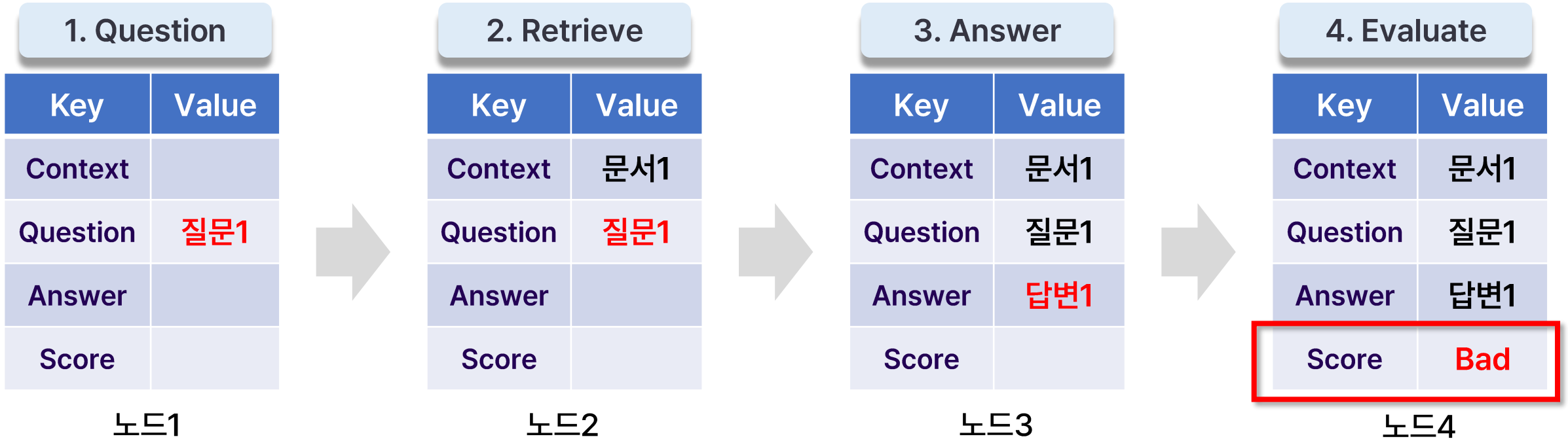
LangGraph를 활용하면?



상태(State)

: 노드와 노드 간 정보 전달 시 State 객체에 담아 전달

Dictionary 형태와 유사하고 새로운 노드에 값을 덮어쓰는 방식(Overwrite)으로 진행



노드1: 질문 재작성 요청

5. Question

Key	Value
Context	문서1
Question	질문2
Answer	답변1
Score	Bad

노드1

6. Retrieve

Key	Value
Context	문서2
Question	질문2
Answer	답변1
Score	Bad

노드2

7. Answer

Key	Value
Context	문서2
Question	질문2
Answer	답변2
Score	Bad

노드3

8. Evaluate

Key	Value
Context	문서2
Question	질문2
Answer	답변2
Score	Good

노드4

Question Transform: 질문 재작성

노드2: 문서 검색을 재요청

1. Question

Key	Value
Context	
Question	질문1
Answer	
Score	

노드1

5. Retrieve

Key	Value
Context	문서2
Question	질문1
Answer	답변1
Score	Bad

노드2

6. Answer

Key	Value
Context	문서2
Question	질문1
Answer	답변2
Score	Bad

노드3

7. Evaluate

Key	Value
Context	문서2
Question	질문1
Answer	답변2
Score	Good

노드4

Context Retrieval 조정
(Chunk, 다른 검색기, Web Search, ...)

노드3: 답변 재생성 요청

1. Question

Key	Value
Context	
Question	질문1
Answer	
Score	

노드1

2. Retrieve

Key	Value
Context	문서1
Question	질문1
Answer	
Score	

노드2

5. Answer

Key	Value
Context	문서1
Question	질문1
Answer	답변2
Score	Bad

노드3

6. Evaluate

Key	Value
Context	문서1
Question	질문1
Answer	답변2
Score	Good

노드4

프롬프트 조정 & 다른 LLM 사용
(GPT, Claude, ...)