LLM(대규모 언어 모델, Large Language Model)

LLM은 Large Language Model(대규모 언어 모델)의 약자로, 방대한 양의 텍스트 데이터로 학습된 인공지능 모델입니다.

주요 활용 분야

LLM은 질문 응답, 문서 작성 및 편집, 프로그래밍 코드 생성, 다국어 번역, 텍스트 요약, 데이터 분석 지원, 창의적 콘텐츠 제작 등 다양한 분야에서 활용됩니다.

LLM의 단점과 한계

1. 진짜 이해 부족
    - LLM은 단어 패턴을 예측하는 모델일 뿐, 사람처럼 내용을 “이해”하지 않습니다.
    - 그래서 겉으로는 똑똑해 보이지만, 실제로는 통계적 추론에 불과합니다.
2. 사실 오류(Hallucination)
    - 존재하지 않는 정보도 그럴듯하게 만들어낼 수 있습니다.
    - 예: 실제로 없는 책 제목이나 잘못된 역사적 사실을 말하기도 함.

3. 최신 정보 부족
    - 학습한 시점 이후의 새로운 사건이나 데이터는 알 수 없습니다.
    - 예: 모델이 2023년까지 학습했다면, 2025년 사건은 모를 수 있음.
4. 편향(Bias) 문제
    - 학습 데이터에 들어 있는 사회적 편견이나 차별적 표현을 그대로 따라 할 수 있습니다.
    - 예: 특정 성별, 직업, 문화에 대한 고정관념을 답변에 반영할 수 있음.
5. 고비용·자원 소모
    - LLM을 학습하거나 실행하는 데는 엄청난 컴퓨팅 자원과 에너지가 필요합니다.
    - 환경 부담이나 사용 비용이 큰 단점입니다.

RAG - Retrieval Augmented Generation

RAG는 검색 증강 생성이라는 뜻으로, LLM의 단점을 보완하기 위해 개발된 기술입니다.
간단히 말하면, LLM이 답변하기 전에 먼저 관련 자료를 검색해서 참고하도록 만든 방식입니다.

작동 원리

1단계 - 검색(Retrieval)

사용자가 질문하면, 먼저 데이터베이스나 문서 저장소에서 관련된 정보를 찾습니다.

2단계 - 증강(Augmentation)

찾은 정보를 LLM에게 함께 제공합니다. "이 자료를 참고해서 답변해"라고 알려주는 것입니다.

3단계 - 생성(Generation)

LLM이 검색된 실제 자료를 바탕으로 답변을 생성합니다.

LLM 단점을 어떻게 극복하나?

1. **환각 현상 감소**:
   > 실제 문서를 참조하므로 정보를 지어내는 대신 근거 있는 답변을 제공합니다.
2. **최신 정보 제공**:
   > 데이터베이스를 업데이트하면 최신 정보로 답변할 수 있어요. 학습 시점의 제약을 극복합니다.
3. **출처 명시 가능**:
   > 어떤 문서에서 정보를 가져왔는지 보여줄 수 있어 신뢰성이 높아집니다.
4. **특정 도메인 지식**:
   > 회사 내부 문서, 전문 자료 등 특화된 정보를 활용할 수 있어요.

실제 활용 예시

- **기업 지식 관리**:
  - 회사의 매뉴얼, 보고서, 정책 문서를 검색해서 직원 질문에 답변합니다.
- **고객 지원**:
  - 제품 설명서와 FAQ를 참조해서 고객 문의에 정확하게 응답합니다.
- **의료 정보**:
  - 최신 의학 논문과 가이드라인을 검색해서 근거 기반 정보를 제공합니다.
- **법률 자문**:
  - 법률 문서와 판례를 찾아 법적 질문에 답변합니다.

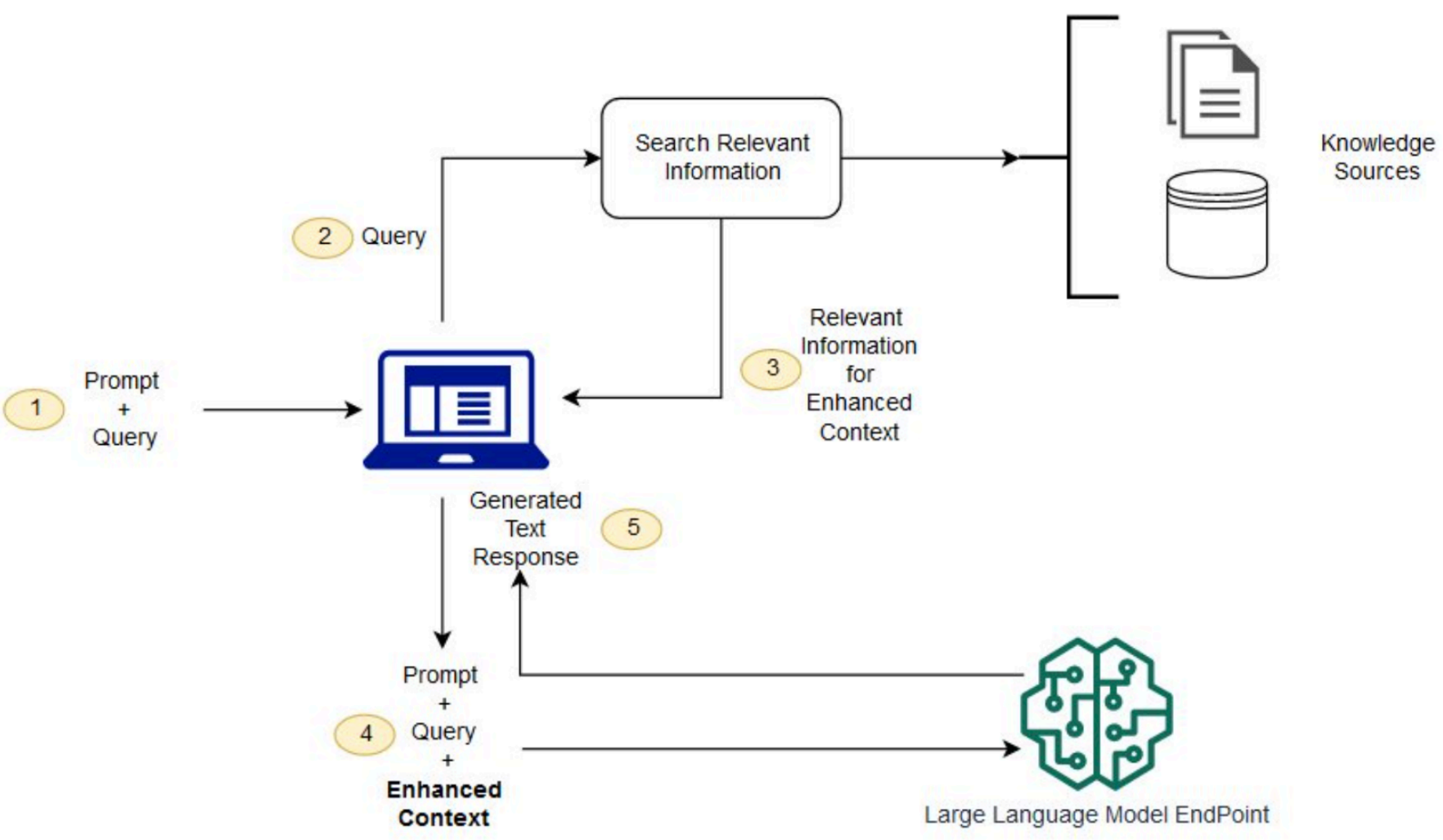
RAG의 장점

- 정확도가 높고, 검증 가능하며, 업데이트가 쉽고, 비용 효율적입니다.
- 전체 모델을 재학습할 필요 없이 문서만 추가하면 되기 때문이에요.

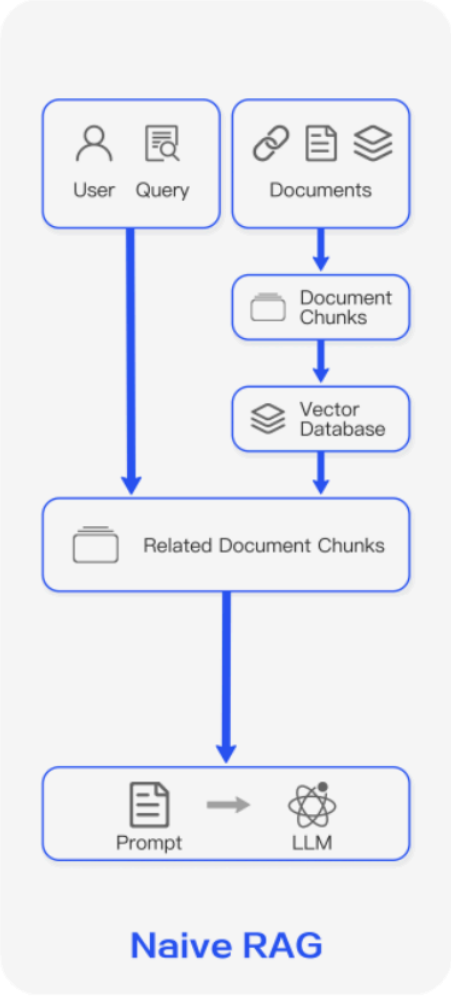
RAG의 한계

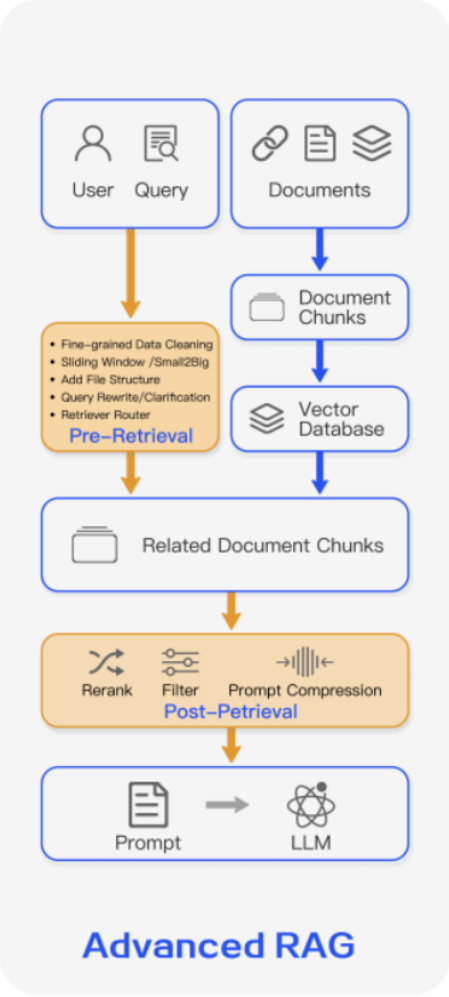
- 완벽하지는 않습니다.
- 검색이 잘못되면 여전히 부정확한 답변이 나올 수 있고, 문서에 없는 내용은 답변하기 어려우며, 검색 시간 때문에 응답이 조금 느릴 수 있습니다.

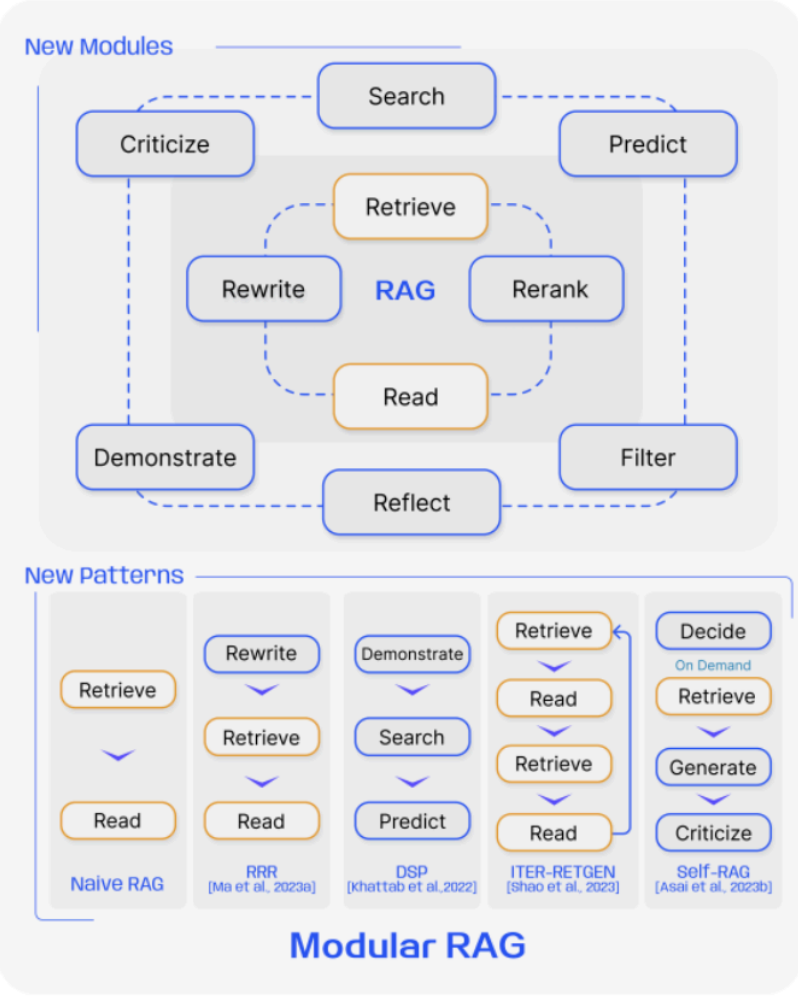
RAG 워크플로우


Search Relevant Information
Knowledge Sources
2 Query
1 Prompt + Query
3 Relevant Information for Enhanced Context
Generated Text Response
5
4 Prompt + Query + Enhanced Context
Large Language Model EndPoint

RAG의 3가지 주요 패러다임







1. Naive RAG (기초형 RAG)

> 학생이 검색 결과를 복사해서 답안에 그대로 옮겨 쓰는 느낌.

항목	내용
구조	단순한 `Retrieval → Generation` 파이프라인
특징	검색된 문서를 그대로 LLM에 넣고 답변 생성
장점	구현이 간단하고 빠름
단점	검색 품질, 문서 길이, 중복 정보에 취약함
예시	LangChain 기본 RAG, simple vector search 기반

2. Advanced RAG (고도형 RAG)

> 학생이 검색 결과를 요약·정리해서 자기 말로 답을 쓰는 느낌.

항목	내용
구조	Retrieval → **Preprocessing (정제)** → Generation
특징	문서 요약, 중복 제거, 랭킹 조정, Query Rewriting 등
장점	더 관련성 높은 정보 제공, 답변 품질 향상
단점	처리 단계가 많아 속도 느림
예시	Self-RAG, Adaptive RAG, Corrective RAG 등

3. Modular RAG (모듈형 RAG)

> 학생이 검색 담당, 요약 담당, 검토 담당 등 역할을 나누어 협업하는 팀처럼 작동.

항목	내용
구조	Retrieval + Generation + Feedback + Memory 등 **모듈 단위 구성**
특징	각 기능(검색, 필터링, 압축, 평가)을 별도 모듈로 관리
장점	유지보수 용이, 실험 확장성 높음, 멀티모달 통합 가능
단점	설계 복잡도 높고 구현 난이도 있음
예시	LangGraph 기반 RAG, GraphRAG, Modular pipeline RAG

세 가지 RAG 비교 요약

구분	Naive RAG	Advanced RAG	Modular RAG
구조	단일 파이프라인	정제 및 피드백 포함	모듈 단위 조합
복잡도	★☆☆☆☆	★★★☆☆	★★★★★
품질	중간	높음	최고 (확장성 높음)
속도	빠름	중간	느림
예시	LangChain 기본형	Self-RAG, Corrective-RAG	LangGraph, GraphRAG