2026/08/28, AI 에이전트 세미나

이론부터 실전까지 AI 에이전트 완벽 마스터

- 6장 정보 검색과 증강을 위한 고급 RAG 기법 -

손 우 영(wooyoung@pel.sejong.ac.kr)

세종대학교 프로토콜공학연구실

목 차

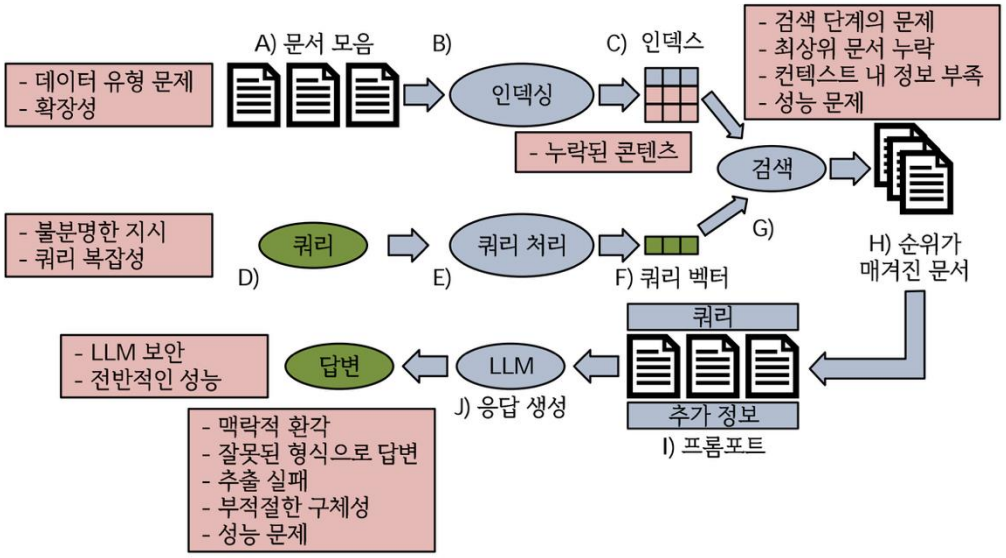
- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

고급 RAG

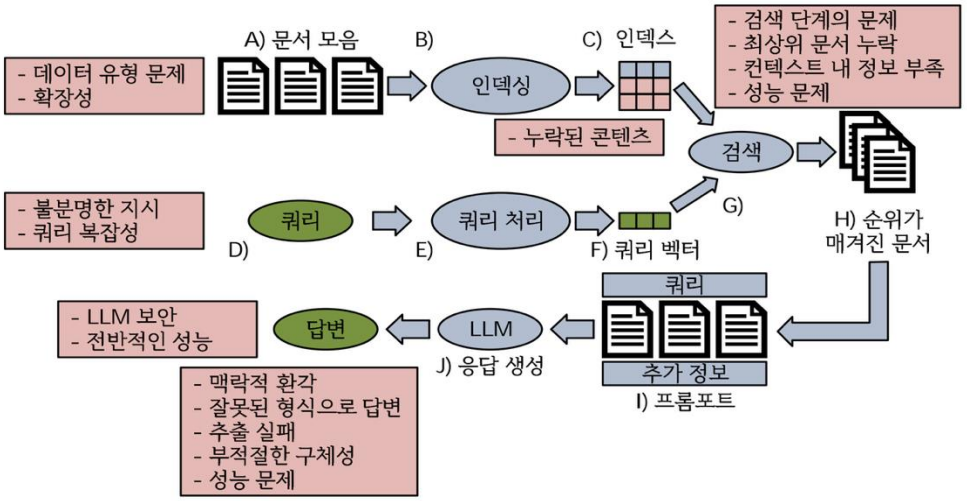
- 나이브 RAG의 한계점 (1/3)
  - 쿼리 처리 과정
    - 불명확한 지시나 복잡 질의에서는 사용자의 검색 의도와 핵심 조건을 정확히 해석하기 어려움
    - 쿼리 분해·재작성·라우팅이 부정확하면 핵심 정보가 누락되거나 잘못된 검색 경로가 선택될 수 있음



(그림 1) 나이브 RAG의 문제 요약 및 파이프라인 내 단계별 문제 발생 지점 식별

고급 RAG

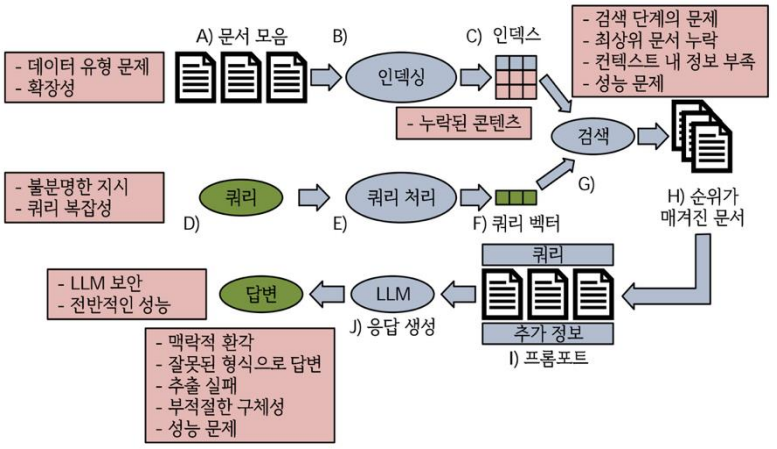
- 나이브 RAG의 한계점 (2/3)
  - 검색 과정
    - 정밀도와 재현율을 동시에 확보하기 어려우며, Top-k 검색 과정에서 정답에 필요한 문서가 누락될 수 있음
    - 지식 베이스와 인덱스가 오래되거나 청킹 품질이 낮으면 관련성이 떨어지는 정보, 중복·상충 정보가 검색될 수 있음
    - 컨텍스트 길이 제한으로 검색된 정보를 LLM에 모두 전달하지 못할 수 있음



(그림 1) 나이브 RAG의 문제 요약 및 파이프라인 내 단계별 문제 발생 지점 식별

고급 RAG

- 나이브 RAG의 한계점 (3/3)
  - 생성 과정
    - 올바른 컨텍스트를 제공해도 노이즈나 상충 정보로 인해 핵심 근거 추출에 실패하거나 맥락적 할루시네이션이 발생할 수 있음
    - 요구한 답변 형식·구체성·완전성을 충족하지 못하거나 일부 항목을 누락할 수 있음
    - 검색 결과의 출처와 신뢰성을 충분히 검증하지 못하면 부정확한 답변으로 이어질 수 있음



(그림 1) 나이브 RAG의 문제 요약 및 파이프라인 내 단계별 문제 발생 지점 식별

고급 RAG

- 개념
  - 나이브 RAG의 한계를 보완하기 위해 검색 전·검색·검색 후 단계의 구성 요소를 세밀하게 최적화한 RAG 방식

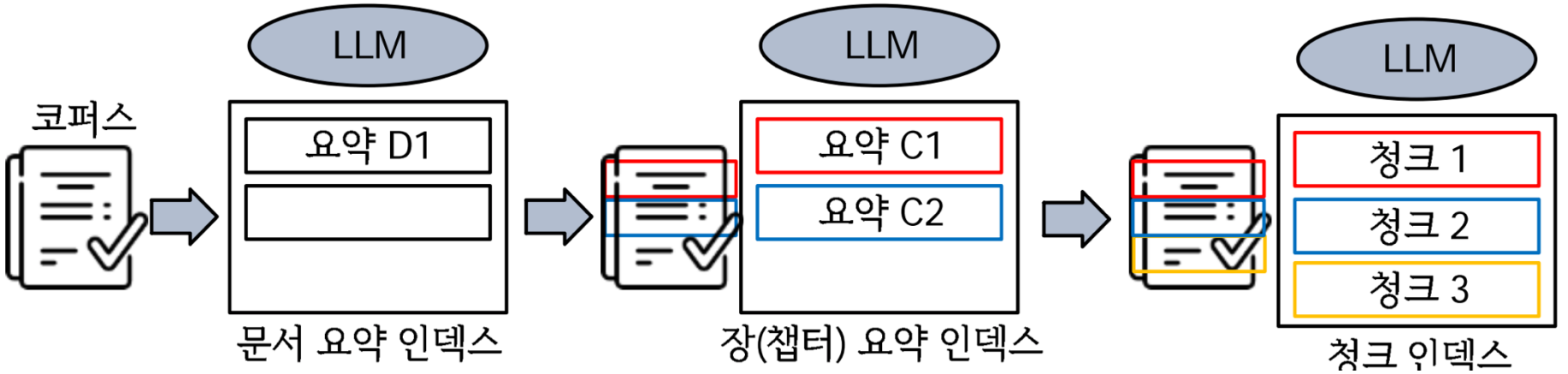
- 도입 배경
  - 나이브 RAG는 모든 청크를 동일한 수준에서 검색하므로 문서·섹션·청크 사이의 계층 구조를 반영하기 어려움
  - 문서의 양이 증가하면 검색 지연이 커지고, 긴 문서에서는 관련 청크를 정확하게 찾기 어려워짐

고급 RAG

- 계층적 인덱싱 (1/5)
  - 개요
    - 여러 장과 섹션으로 구성된 긴 문서는 먼저 관련 장을 찾고, 해당 장 내부에서 필요한 섹션을 검색하는 단계적 접근이 효과적임
    - 나이브 RAG의 평면적 청킹 방식은 문서의 계층 구조를 반영하지 못하며, 큰 단위의 텍스트는 크기가 크고 노이즈가 많이 맥락적 중요성을 정확히 표현하기 어려움
  - 개념
    - 문서를 문서-장·섹션-청크 단위로 계층화하고, 각 계층의 요약과 메타데이터를 임베딩하여 인덱싱하는 방식임

고급 RAG

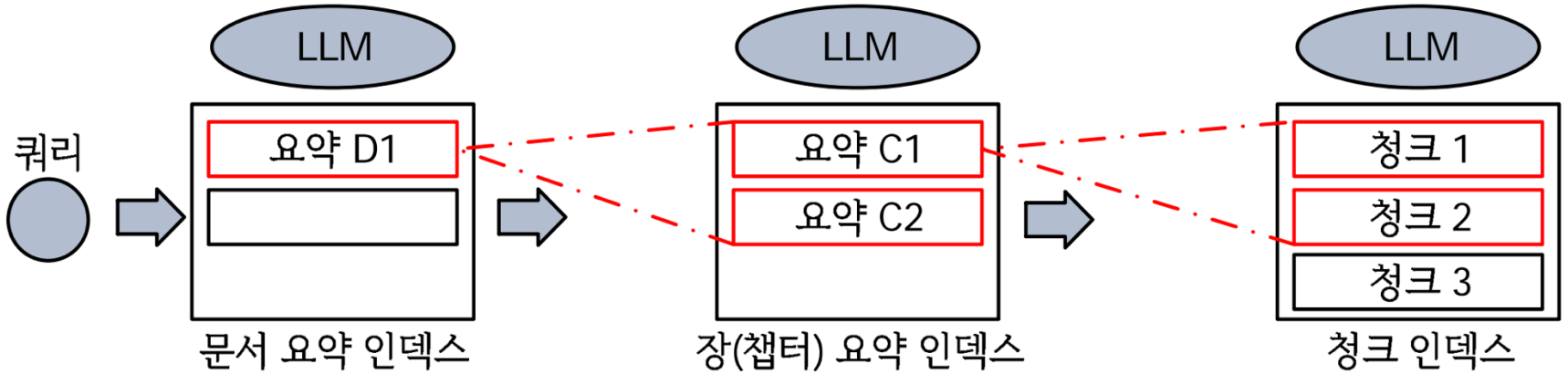
- 계층적 인덱싱 (2/5)
  - 인덱싱 생성 과정
    1. 코퍼스를 문서 단위로 구분하고 각 문서의 핵심 내용을 나타내는 문서 요약을 생성함
    2. 문서 내부의 장·섹션·제목·소제목에도 더 세밀한 요약을 생성하여 하위 계층을 구성함
    3. 생성된 요약은 임베딩하고, 원본 문서 및 하위 청크와의 관계를 메타데이터로 저장함



(그림 2) 계층적 인덱싱의 인덱싱 생성 과정

고급 RAG

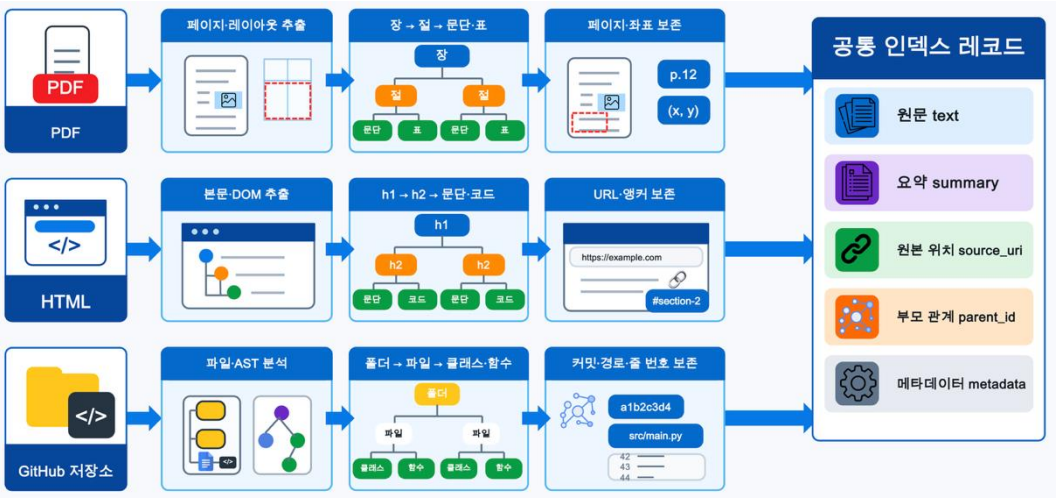
- 계층적 인덱싱 (3/5)
  - 검색 과정
    1. 쿼리와 문서 요약 임베딩의 유사도를 계산하여 우선 관련 문서를 선택함
    2. 선택된 문서의 장·섹션 요약을 다시 검색한 뒤, 최종적으로 관련성이 높은 청크를 탐색함
    3. 메타데이터를 이용하여 상위 요약과 연결된 실제 원문 청크를 LLM에 제공함



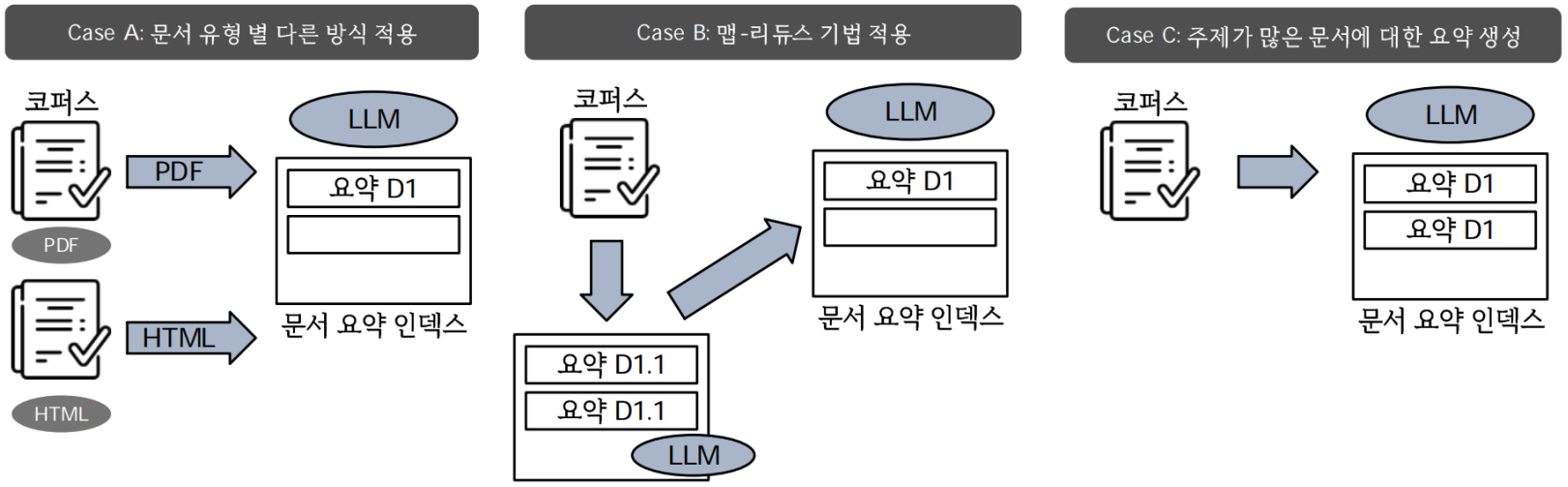
(그림 3) 계층적 인덱싱의 검색 과정

고급

- 계층적 인덱싱 (4/5)
  - 계층적 인덱싱의 변형 (1/2)
    - 문서 유형별 처리
      - PDF·HTML·GitHub 저장소 등 문서 유형에 따라 서로 다른 분할 방식을 적용함
      - 파일 구조에 맞춰 요약과 임베딩을 생성하므로 일종의 텍스트 정규화 역할을 수행함



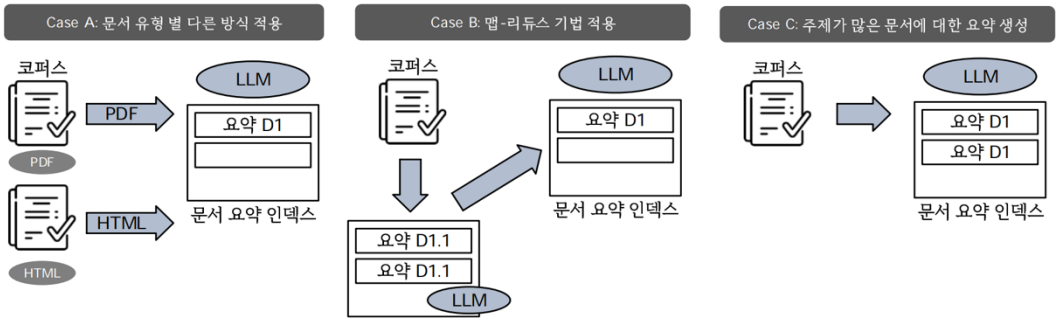
(그림 5) PDF·HTML·GitHub 저장소의 구조별 파싱 과정



(그림 4) 계층적 인덱싱의 변형

고급 RAG

- 계층적 인덱싱 (5/5)
  - 계층적 인덱싱의 변형 (2/2)
    - 맵리듀스 기반 요약
      - 문서가 너무 길어 한 번에 요약하기 어려운 경우 여러 부분으로 나누어 중간 요약을 생성함
      - 생성된 중간 요약을 다시 결합하여 문서 전체를 대표하는 최종 요약을 생성함
    - 복수 요약 생성
      - 하나의 문서가 지나치게 많은 주제를 다루는 경우 문서마다 여러 개의 요약을 생성함
      - 일정 토큰 또는 페이지 단위로 요약을 분리하여 의미적 노이즈가 검색 결과에 미치는 영향을 줄임



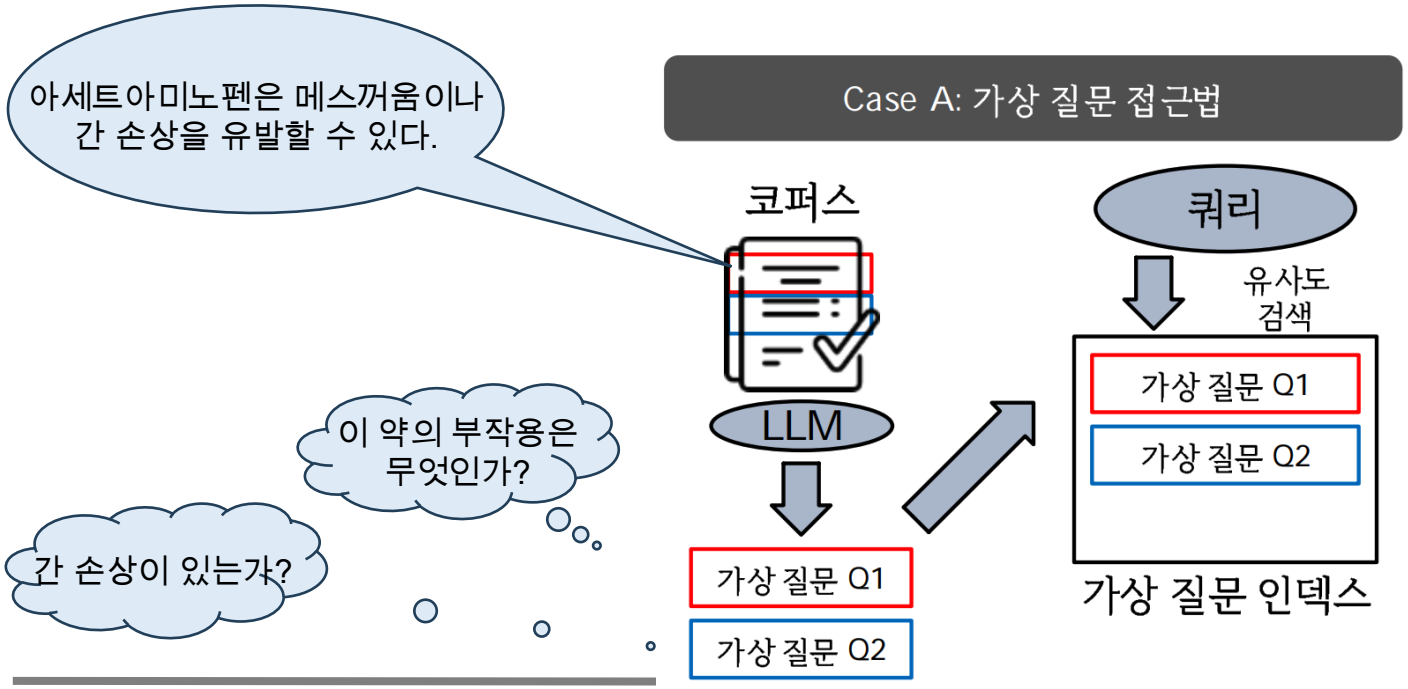
(그림 4) 계층적 인덱싱의 변형

고급 RAG

- 가상 질문 접근법과 HyDE
  - 개요
    - 나이브 RAG는 사용자 쿼리와 원본 청크의 임베딩을 직접 비교하므로 질문 형태의 쿼리와 설명 형태의 문서 사이에 의미적 차이가 발생할 수 있음
    - 사용자의 질문 방식과 시스템의 활용 목적을 반영하여 쿼리와 검색 대상 사이의 의미적 유사성을 높일 필요가 있음
  - 핵심 아이디어
    - 원본 청크를 직접 검색하는 대신 가상의 질문 또는 답변을 생성하여 쿼리와 청크 사이의 의미적 연결 고리를 구성함
    - 가상 질문은 '청크 → 예상 질문'을 생성하고, HyDE는 '사용자 쿼리 → 가상 답변'을 생성한다는 차이가 있음

고급 RAG

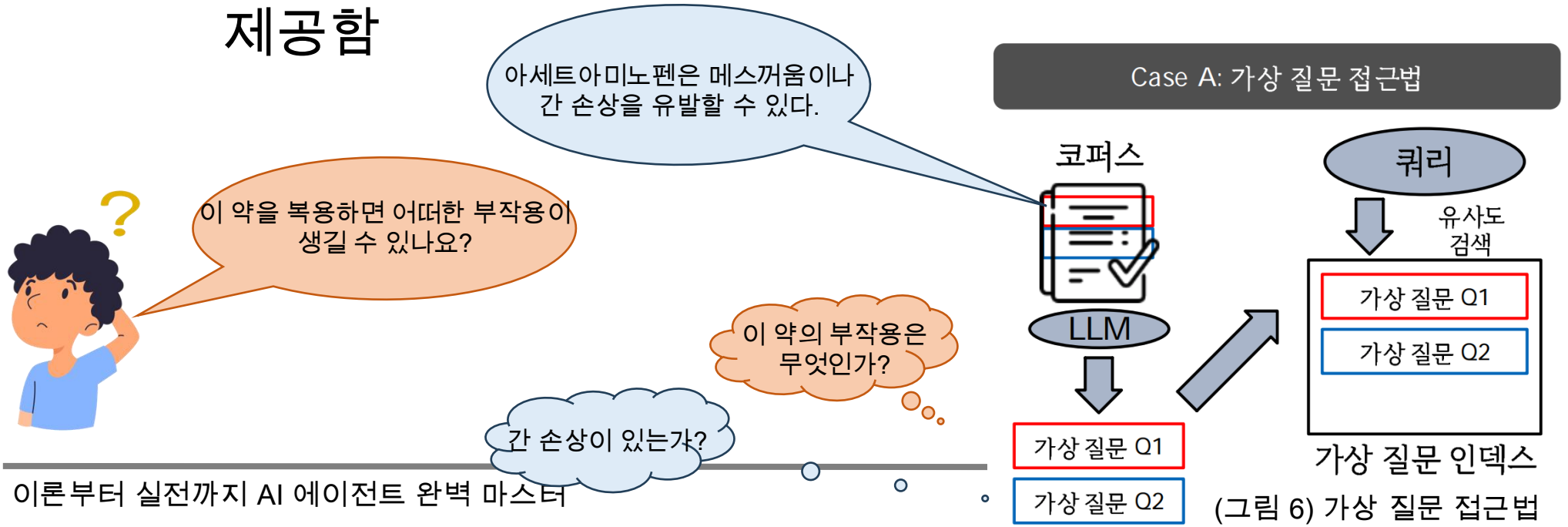
- 가상 질문 접근법 (1/3)
  - 검색 전 단계
    1. LLM을 이용하여 각 청크가 답변할 수 있는 하나 이상의 가상 질문을 생성함
    2. 생성된 가상 질문을 임베딩하여 벡터 데이터베이스에 저장하고, 원본 청크와의 연결 관계를 메타데이터로 관리함



(그림 6) 가상 질문 접근법

고급 RAG

- 가상 질문 접근법 (2/3)
  - 쿼리 단계
    1. 사용자 쿼리를 임베딩한 뒤 저장된 가상 질문 벡터와 유사도를 비교함
    2. 사용자 쿼리와 가장 유사한 가상 질문을 찾고, 메타데이터를 통해 해당 질문이 생성된 원본 청크를 확인함
    3. 검색된 원본 청크를 LLM의 답변 생성을 위한 컨텍스트로 제공함



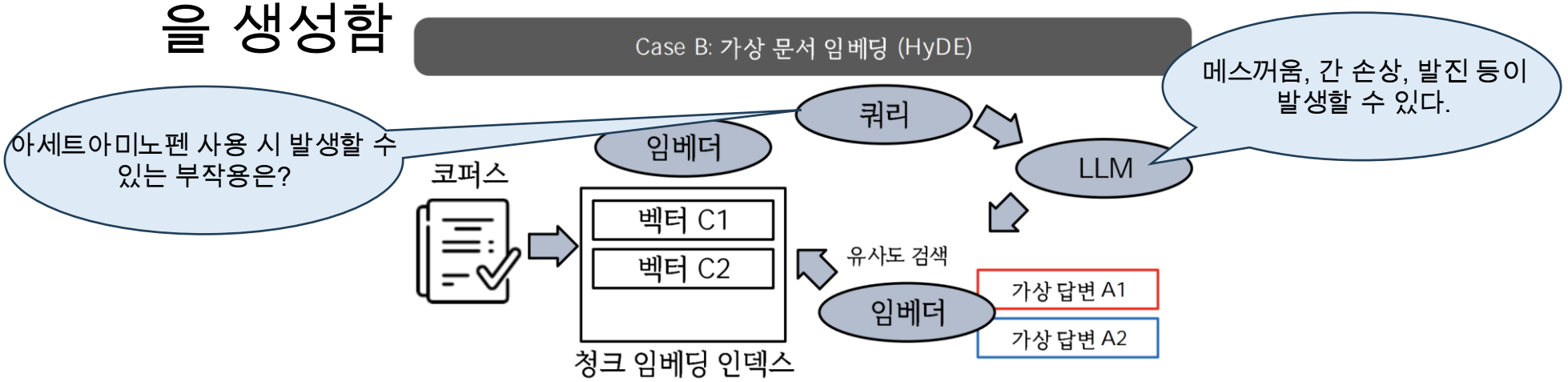
(그림 6) 가상 질문 접근법

고급 RAG

- 가상 질문 접근법 (3/3)
  - 장점
    - 질문 형태의 쿼리와 가상 질문을 직접 비교하므로 원본 청크를 검색하는 것보다 의미적 유사성을 높일 수 있음
    - 하나의 청크에 여러 개의 가상 질문을 만드는 경우, 다양한 질문 표현을 포괄할 수 있음
  - 한계점
    - 청크마다 여러 개의 가상 질문을 생성하면 다양한 표현을 포괄할 수 있지만 생성·임베딩·저장 비용도 증가함

고급 RAG

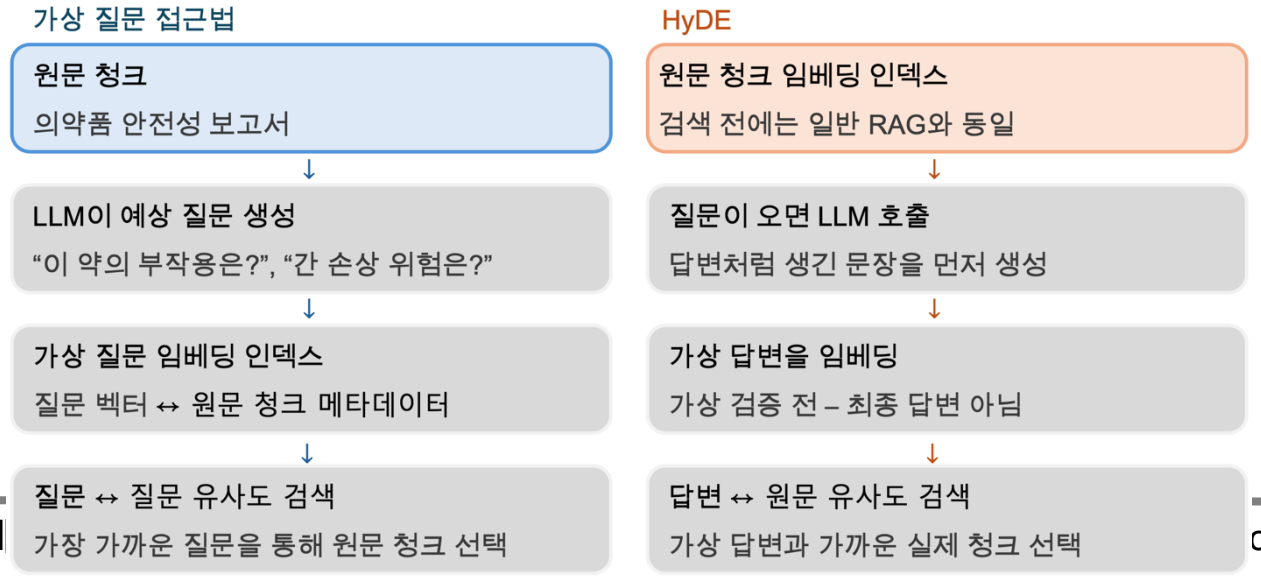
- HyDE (Hypothetical Document Embedding)
  - 동작 과정
    1. 사용자 쿼리가 입력되면 LLM이 검색을 수행하지 않은 상태에서 쿼리에 대한 가상 답변을 먼저 생성함
    2. 생성된 가상 답변을 임베딩하고, 해당 벡터와 가장 유사한 원본 청크를 벡터 데이터베이스에서 검색함
    3. 검색된 청크를 실제 근거 컨텍스트로 사용하여 최종 답변을 생성함



(그림 7) HyDE 접근 방식

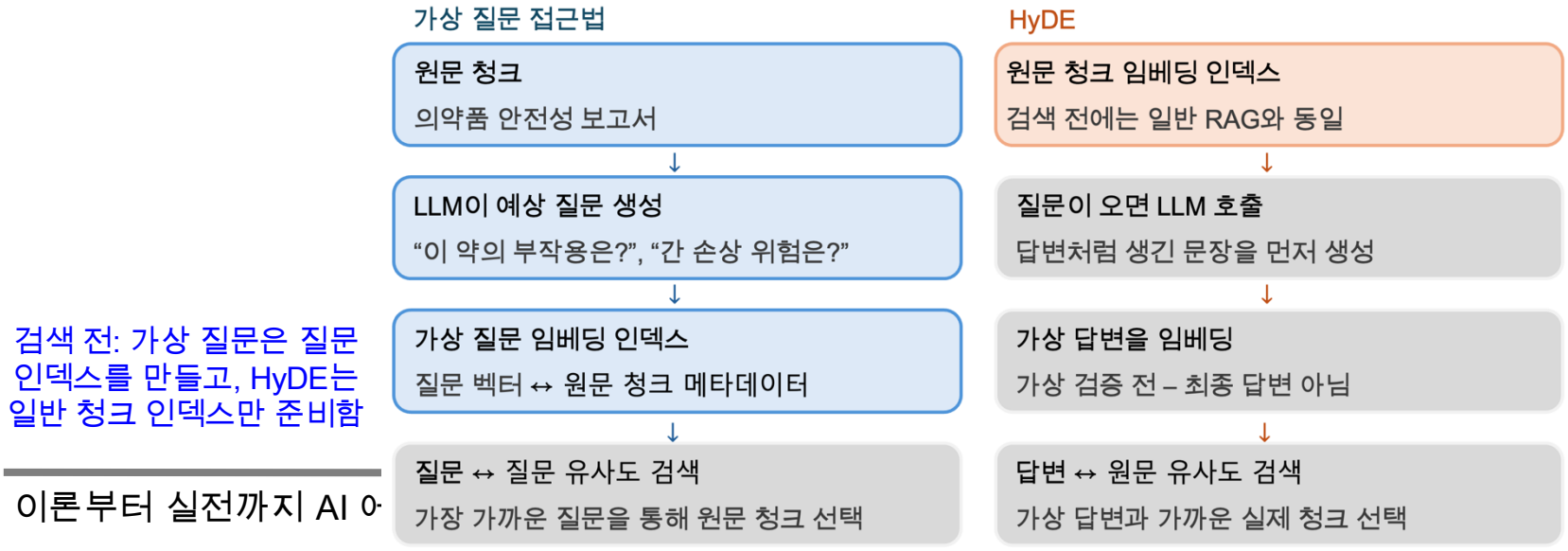
고급 RAG

- 가상 질문과 HyDE (1/5)
  - 검색 전 단계 비교 (1/2)
    - 가상 질문 접근법
      - 문서를 청크 단위로 분할하고, 각 청크에 대해 예상 가능한 질문을 미리 생성·임베딩하여 가상 질문 인덱스를 구축함
    - HyDE
      - 별도의 가상 질문 인덱스를 만들지 않고 기존 RAG와 동일하게 원본 청크의 임베딩 인덱스를 사용함



고급 RAG

- 가상 질문과 HyDE (2/5)
  - 검색 전 단계 비교 (2/2)
    - 가상 질문 접근법
      - 문서를 청크 단위로 분할하고, 각 청크에 대해 예상 가능한 질문을 미리 생성·임베딩하여 가상 질문 인덱스를 구축함
    - HyDE
      - 별도의 가상 질문 인덱스를 만들지 않고 기존 RAG와 동일하게 원본 청크의 임베딩 인덱스를 사용함



고급 RAG

- 가상 질문과 HyDE (3/5)
  - 쿼리 단계 비교 (1/2)
    - 가상 질문 접근법
      - 사용자 쿼리와 가상 질문 벡터를 비교하고, 가장 유사한 질문에 연결된 원본 청크를 검색함
    - HyDE
      - 사용자 쿼리로부터 “메스꺼움·간 손상·발진 등이 발생할 수 있다”와 같은 가상 답변을 생성함
      - 가상 답변을 임베딩하고, 해당 벡터와 가장 유사한 원본 청크를 검색함

사용자 질문: “아세트아미노펜 사용 시 발생할 수 있는 부작용은?”

질문 시점: 가상 질문은 저장된 질문과 비교하고,
HyDE는 LLM으로 가상 답변을 만듦

고급 RAG

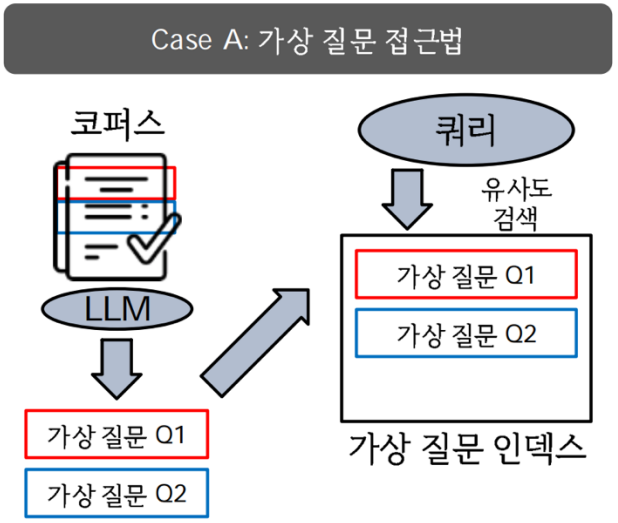
- 가상 질문과 HyDE (4/5)
  - 쿼리 단계 비교 (2/2)
    - 가상 질문 접근법
      - 사용자 쿼리와 가상 질문 벡터를 비교하고, 가장 유사한 질문에 연결된 원본 청크를 검색함
    - HyDE
      - 사용자 쿼리로부터 “메스꺼움·간 손상·발진 등이 발생할 수 있다”와 같은 가상 답변을 생성함
      - 가상 답변을 임베딩하고, 해당 벡터와 가장 유사한 원본 청크를 검색함

사용자 질문: “아세트아미노펜 사용 시 발생할 수 있는 부작용은?”

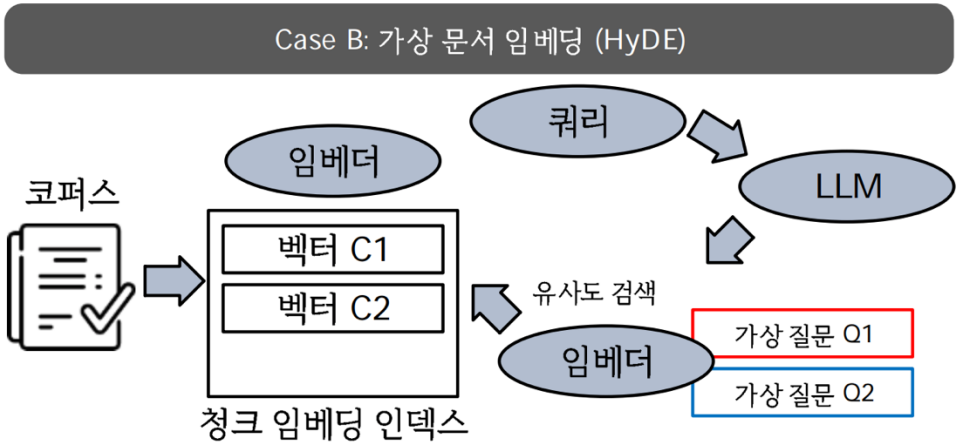
유사도 검색: 질문↔질문과
가상 답변↔원문이라는
서로 다른 비교를 수행함

고급 RAG

- 가상 질문과 HyDE (5/5)
  - 핵심 차이
    - 가상 질문은 인덱싱 단계에서 청크로부터 질문을 생성하는 방식임
    - HyDE는 쿼리 단계에서 질문으로부터 답변을 생성하는 방식임



(그림 6) 가상 질문 접근법



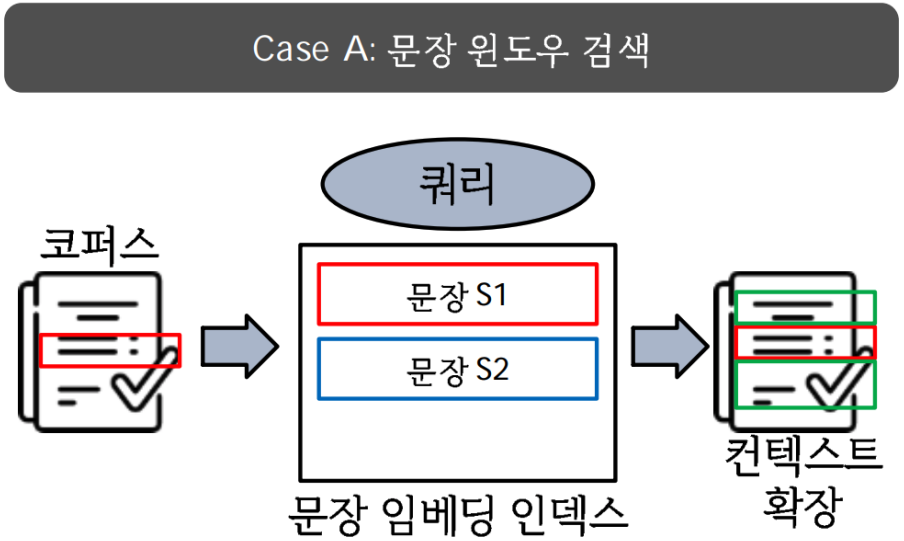
(그림 7) HyDE 접근 방식

고급 RAG

- 컨텍스트 강화 (1/5)
  - 개요
    - 작은 청크는 쿼리의 세부 내용을 정밀하게 검색할 수 있지만, LLM의 답변 생성에 필요한 주변 맥락이 누락될 수 있음
    - 큰 청크는 풍부한 정보를 제공하지만 불필요한 내용과 노이즈가 포함되어 검색 정확도가 낮아질 수 있음
  - 핵심 원리
    - 작은 단위로 관련 정보를 정밀하게 검색한 뒤, 검색된 정보의 주변 문장이나 상위 문서를 추가하여 LLM에 제공할 컨텍스트를 확장함
    - 검색 단계에서는 작은 청크의 정밀도를 활용하고, 생성 단계에서는 더 넓은 맥락을 제공하여 검색 품질과 생성 품질의 균형을 맞춤

고급 RAG

- 컨텍스트 강화 (2/5)
  - 문장 윈도우 검색 (1/2)
    - 동작 과정
      1. 문서를 문장 단위로 분할하고 각 문장을 개별적으로 임베딩하여 문장 임베딩 인덱스를 구성함
      2. 사용자 쿼리와 가장 유사한 문장 x를 검색한 뒤, 해당 문장의 앞뒤 k개 문장을 함께 가져와 컨텍스트를 확장함

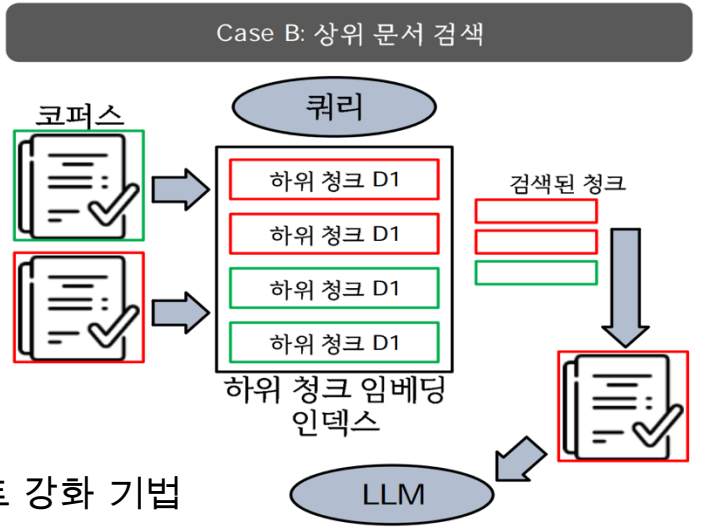


(그림 8) 컨텍스트 강화 기법

고급 RAG

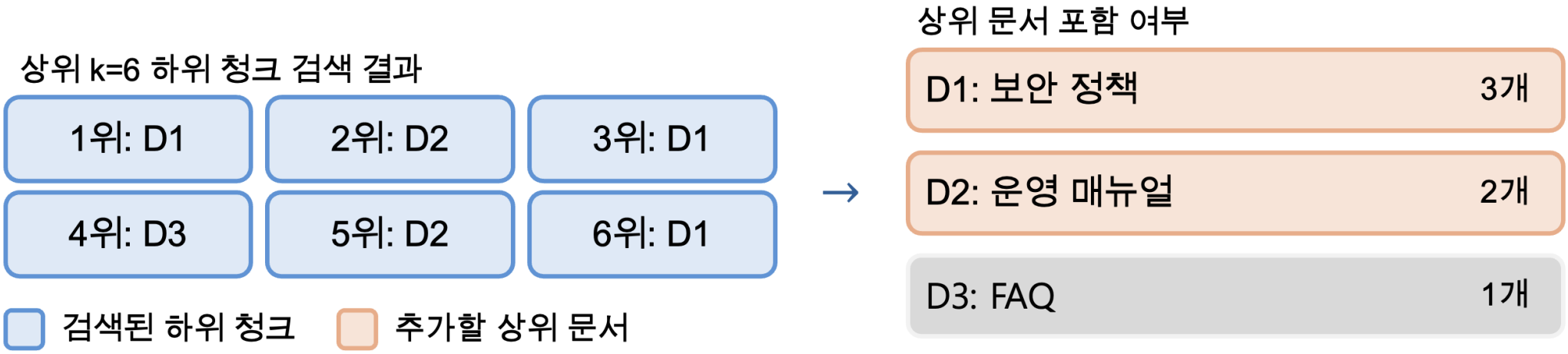
- 컨텍스트 강화 (3/5)
  - 문장 윈도우 검색 (2/2)
    - 특징
      - 문장 단위의 작은 임베딩을 사용하므로 쿼리의 세부 사항과 직접 관련된 정보를 정밀하게 검색할 수 있음
      - 검색된 문장의 주변 내용을 함께 제공하므로 문장 단위 검색에서 발생하는 맥락 손실을 보완할 수 있음
    - 윈도우 설정 기준
      - k 가 너무 작으면 답변에 필요한 맥락이 누락될 수 있고, 너무 크면 불필요한 문장과 노이즈가 증가할 수 있음
      - 문서의 문장 밀도와 질문에 필요한 정보 범위를 고려하여 적절한 윈도우 크기를 설정해야 함

고급 RAG



(그림 8) 컨텍스트 강화 기법

- 컨텍스트 강화 (4/5)
  - 상위 문서 검색 (1/2)
    - 동작 과정
      1. 문서를 작은 하위 청크로 분할하여 임베딩하되, 각 청크가 속한 상위 문서와의 계층 관계를 메타데이터로 유지함
      2. 쿼리와 관련된 하위 청크를 먼저 검색한 뒤, 해당 청크가 속한 상위 문서를 찾아 LLM의 컨텍스트로 제공함



(그림 9) 상위 k개 하위 청크의 문서별 출현 빈도에 따른 상위 문서 선택

고급 RAG

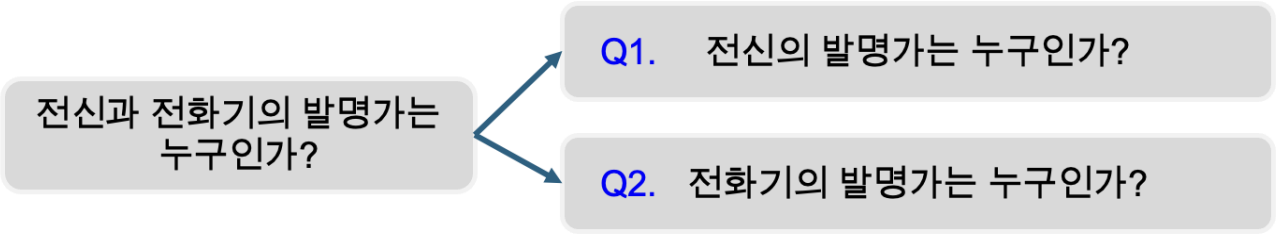
- 컨텍스트 강화 (5/5)
  - 상위 문서 검색 (2/2)
    - 특징
      - 작은 하위 청크를 이용한 정밀 검색과 상위 문서가 제공하는 풍부한 맥락을 함께 활용할 수 있음
      - 여러 문장이나 섹션에 분산된 정보를 종합해야 하는 질문에 효과적이지만, 상위 문서를 지나치게 많이 추가하면 컨텍스트 길이와 노이즈가 증가할 수 있음
      - 검색 정확도와 컨텍스트 크기의 균형을 고려하여 k와 n을 설정해야 함
    - 상위 문서 선택 기준
      - 검색된 상위 k 개의 하위 청크 중 동일한 상위 문서에서 파생된 청크가 n개 이상인 경우에만 해당 문서를 컨텍스트에 추가함
      - 일부 하위 청크만 관련된 상위 문서가 무분별하게 추가되는 것을 방지하여 컨텍스트의 양을 조절함

고급 RAG

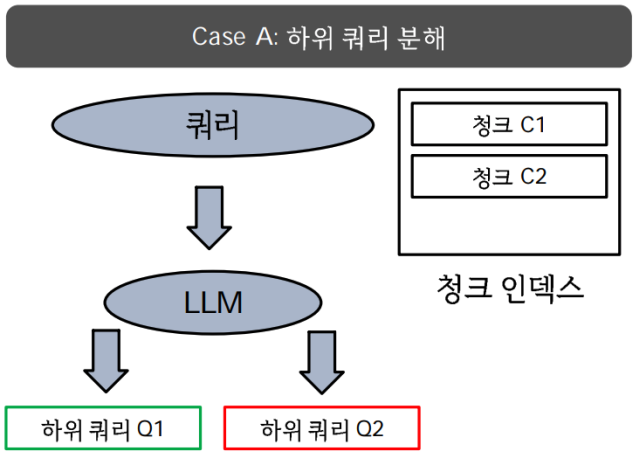
- 쿼리 변환 (1/5)
  - 개념
    - LLM을 활용하여 사용자 쿼리를 변형함으로써 검색 성능을 개선하는 기법
  - 특징
    - 복잡한 쿼리를 여러 개로 분해하면 원래 쿼리에 직접 대응하는 청크가 없더라도 하위 쿼리에 대응하는 청크를 더 쉽게 찾을 수 있음
  - 주요 기법
    - 쿼리 분해, 스텝백 프롬프팅, 쿼리 재작성 및 쿼리 확장 등의 방법을 활용함

고급 RAG

- 쿼리 변환 (2/5)
  - 쿼리 분해
    - 개념
      - 여러 대상이나 조건이 포함된 복잡한 쿼리를 각각 독립적으로 검색할 수 있는 여러 개의 하위 쿼리로 분해함



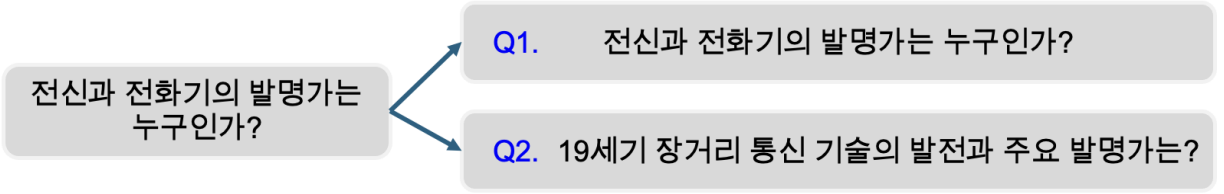
    - 검색 효과
      - 원래 쿼리에 직접 대응하는 청크가 없더라도 분해된 하위 쿼리에 대응하는 청크는 더 쉽게 찾을 수 있음

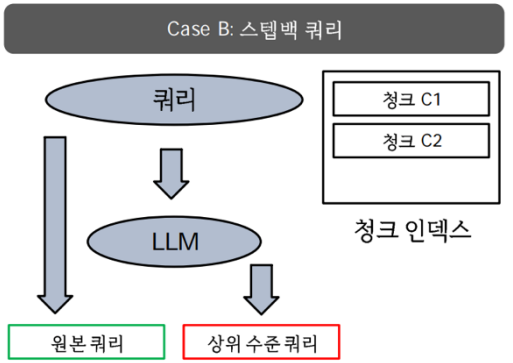


(그림 10) LLM을 활용한 복합 쿼리의 하위 쿼리 분해 과정

고급 RAG

- 쿼리 변환 (3/5)
  - 스텝백 프롬프팅
    - 개념
      - LLM을 이용하여 상위 수준의 컨텍스트와 일치하도록 원래 쿼리보다 더 일반적인 쿼리를 생성함
    - 특징
      - 어려운 문제에서 한 발짝 물러서서 추상화를 통해 고차원적인 원리를 파악하는 인간의 사고 방식에서 비롯됨
      - 생성된 상위 수준의 쿼리와 원래 쿼리를 각각 임베딩하고, 두 쿼리의 검색 결과를 함께 LLM에 제공하여 답변을 생성함

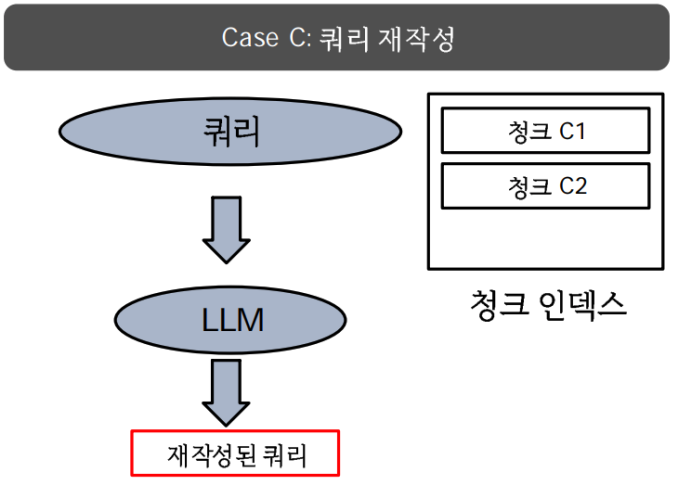




(그림 11) LLM을 활용한 원본 쿼리의 상위 수준 쿼리 생성 과정

고급 RAG

- 쿼리 변환 (4/5)
  - 쿼리 재작성
    - 개념
      - LLM을 이용하여 초기 쿼리를 검색에 더 적합한 형태로 재구성함
    - 특징
      - 질문의 의도는 유지하되, 모호한 표현·구조를 명확하게 정리하고, 핵심 엔티티·관계·조건을 드러낸 하나의 검색 친화적 쿼리로 변환함



(그림 12) LLM을 활용한 검색 친화적 쿼리 재작성 과정

전신과 전화기의 발명가는 누구인가? → Q1. 전신 발명과와 전화기 발명가를 각각 식별하고 근거를 제시하라

고급 RAG

- 쿼리 변환 (5/5)
  - 쿼리 확장
    - 개념
      - 쿼리에 관련 용어를 추가하여 원래 쿼리와 어휘적으로 겹치지 않는 문서까지 검색할 수 있도록 하는 방식
    - 주요 방식
      - LLM이 쿼리에 대한 답변을 생성한 뒤, 해당 답변과 원래 쿼리를 함께 임베딩하여 검색에 활용함
      - 원래 쿼리와 유사한 n개의 쿼리를 생성하고, 생성된 쿼리 집합을 벡터화하여 검색에 활용함
    - 효과 및 한계
      - 쿼리의 모호성을 줄이고 원래는 찾지 못했던 문서를 발견하도록 지원함
      - 관련 없는 문서를 가져올 위험이 있으므로 후처리 기법을 결합하여 관련 문서를 선별하는 것이 바람직함

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (1/7)
  - 키워드 기반 검색
    - 개념
      - 쿼리에 포함된 특정 키워드와 정확히 일치하는 문서를 검색하는 방식
    - 장점
      - 제품명, 회사명 및 특정 업계의 전문 용어와 같이 명확한 용어를 검색하는 데 유리함
    - 단점
      - 오탈자나 동의어가 포함된 쿼리에 취약함
      - 키워드가 사용된 문맥이나 쿼리의 의미를 파악하지 못함

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (2/7)
  - 벡터 기반 검색
    - 개념
      - 쿼리와 문서를 벡터로 표현하여 의미적 유사성을 기준으로 검색하는 방식
    - 장점
      - 쿼리에 포함된 단어가 문서와 정확히 일치하지 않아도 의미가 유사한 문서를 검색할 수 있음
    - 단점
      - 특정 키워드나 전문 용어를 정확하게 찾아내지 못할 수 있음
      - 일부 도메인에서는 의미 검색뿐만 아니라 정확한 키워드 일치가 반드시 필요함

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (3/7)
  - 하이브리드 검색
    - 개념
      - 키워드 기반 검색과 벡터 기반 검색을 결합하여 두 검색 방식의 장점을 모두 활용하는 방식
    - 검색 구성
      - BM25를 이용하여 특정 용어가 포함된 문서를 식별하는 희소 임베딩을 생성함
      - 트랜스포머 기반 모델을 이용하여 쿼리와 문서의 의미를 표현하는 밀집 임베딩을 생성함
      - 최적의 청크를 선택하기 위해 희소 검색과 밀집 검색이 미치는 영향을 균형 있게 조정함

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (4/7)
  - 하이브리드 검색
    - 점수 계산
      - 두 검색 결과의 점수를 가중 조합하여 최종 하이브리드 검색 점수를 계산함
        - $score_{hybrid} = (1-\alpha) \cdot score_{sparse} + \alpha \cdot score_{dense}$
      - α는 0과 1 사이의 값이며, 0은 순수 키워드 검색, 1은 순수 벡터 검색을 의미함
      - 일반적으로 α는 0.4 또는 0.5 정도를 사용하며, 일부 연구에서는 0.3을 제안함

(그림 13) 쿼리 유형에 따른 키워드·벡터·하이브리드 검색 가중치 조정

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (5/7)
  - 활용: 전자제품·패션·가전 등 여러 카테고리에 걸쳐 수백만 개의 상품을 보유한 전자상거래 플랫폼
    - 쿼리 유형 1) 특정 용어 - 브랜드 또는 제품명을 직접 입력하는 쿼리
      - e.g., “iPhone 16”
      - 특징
        - BM25는 “iPhone”과 같은 정확한 키워드를 우선시하여 특정 제품을 검색함

[표 1] ‘iPhone 16’ 쿼리에 대한 키워드·시맨틱·하이브리드 검색 점수 비교

순위	검색 문서	키워드	시멘틱	하이브리드
1	iPhone 16 제품 사양서	0.98	0.76	0.89
2	iPhone 악세사리 모음	0.62	0.42	0.54
3	최신 스마트폰 성능 비교	0.35	0.72	0.50
4	스마트폰 카메라 구매 가이드	0.14	0.57	0.31

시멘틱 검색 비중 α=0.40

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (6/7)
  - 활용: 전자제품·패션·가전 등 여러 카테고리에 걸쳐 수백만 개의 상품을 보유한 전자상거래 플랫폼
    - 쿼리 유형 2) 일반적인 설명: 제품의 특징을 자연어로 설명하는 쿼리
      - e.g., “중간 가격대의 카메라 성능 좋은 스마트폰”
      - 특징
        - 벡터 기반 검색은 “카메라 성능 좋은 스마트폰”과 같은 구문의 의미를 파악하여 검색함

[표 2] ‘중간 가격대의 카메라 성능 좋은 스마트폰’ 쿼리에 대한 키워드·시맨틱·하이브리드 검색 점수 비교

순위	검색 문서	키워드	시멘틱	하이브리드
1	중급형 스마트폰 사진 품질 비교	0.38	0.91	0.78
2	가성비 카메라폰 추천	0.22	0.95	0.77
3	프리미엄 카메라폰 출시 소식	0.44	0.64	0.59
4	중간 가격대 스마트폰 악세서리	0.76	0.31	0.42

시멘틱 검색 비중 α=0.75

고급 RAG

- 키워드 기반 검색과 하이브리드 검색 (7/7)
  - 활용: 전자제품·패션·가전 등 여러 카테고리에 걸쳐 수백만 개의 상품을 보유한 전자상거래 플랫폼
    - 쿼리 유형 3) 혼합된 형태: 조건과 특정 제품명을 함께 사용하는 쿼리
      - e.g., “500달러 이하의 iPhone”
      - 특징
        - 하이브리드 검색은 두 방식을 결합하여 특정 용어, 일반적인 설명 및 혼합형 쿼리를 모두 효과적으로 처리함

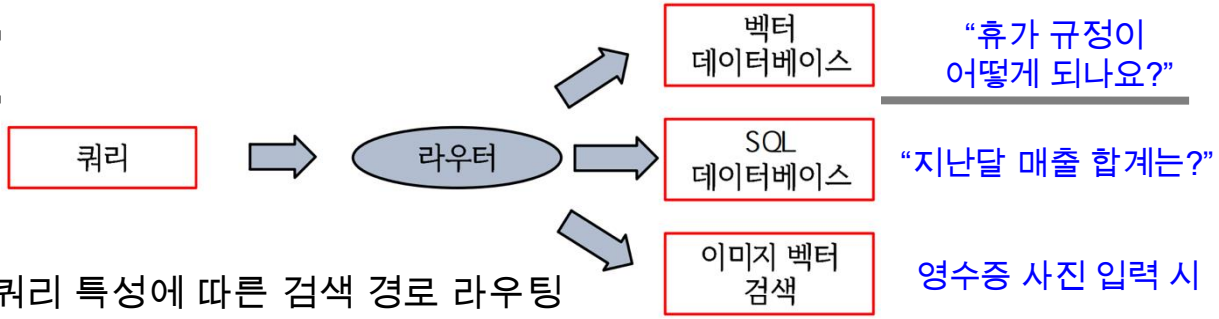
[표 3] ‘500달러 이하의 iPhone’ 쿼리에 대한 키워드·시맨틱·하이브리드 검색 점수 비교

순위	검색 문서	키워드	시멘틱	하이브리드
1	iPhone 보급형 모델 할인 정보	0.91	0.74	0.82
2	500달러 이하 스마트폰 비교	0.64	0.89	0.77
3	중고 iPhone 가격 안내	0.78	0.66	0.72
4	저가 안드로이드폰 추천	0.13	0.77	0.45

시멘틱 검색 비중 α=0.50

고급 RAG

- 쿼리 라우팅 (1/7)
  - 개요
    - 쿼리의 특성에 따라 벡터 데이터베이스, SQL 데이터베이스 및 상용 데이터베이스처럼 서로 다른 검색 대상을 사용해야 할 수 있음
    - 이미지·텍스트·음성과 같은 서로 다른 데이터 소스와 모달리티를 선택해야 하는 경우도 있음
    - 일부 쿼리는 RAG를 사용하지 않고 모델의 parametric 메모리만으로 응답할 수 있음
  - 개념
    - 시스템이 입력된 쿼리에 어떻게 응답할 것인지 결정하고, 적절한 데이터베이스 또는 처리 경로로 쿼리를 전달하는 방식



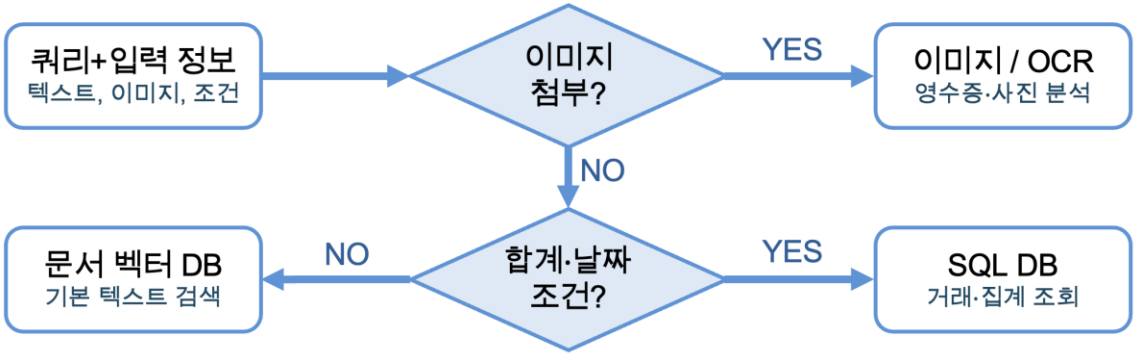
(그림 14) 쿼리 특성에 따른 검색 경로 라우팅

고급 RAG

- 쿼리 라우팅 (2/7)
  - 특징
    - if/else 구문과 유사하지만 하드코딩된 규칙 대신 일반적으로 LLM을 라우터로 사용하여 쿼리마다 경로를 결정함
    - 라우터는 논리적인 규칙 집합 또는 신경망 모델로 구현할 수 있음
    - LLM을 이용한 방식은 비결정론적이므로 항상 올바른 결정을 내리지는 않지만 시스템 성능에 긍정적인 영향을 줄 수 있음

고급 RAG

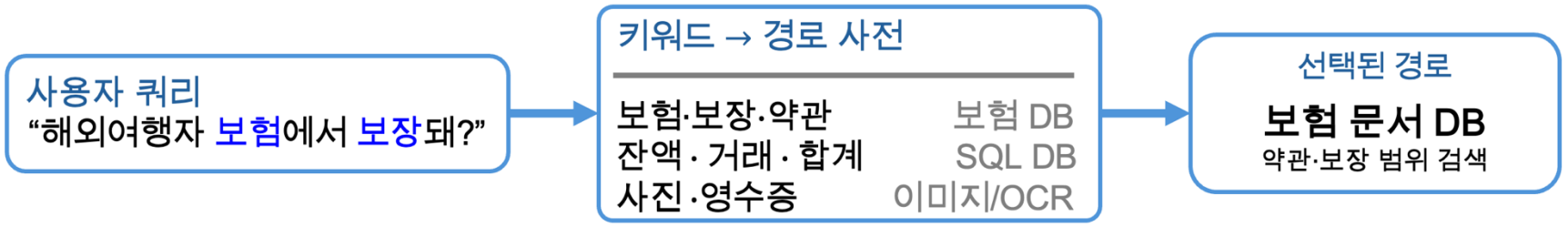
- 쿼리 라우팅 (3/7)
  - 논리적 라우터
    - 개념
      - if/else 조건문으로 구성된 논리 규칙을 이용하여 쿼리의 처리 경로를 결정함
    - 특징
      - 빠르고 결정론적으로 동작하지만 쿼리의 의미를 이해하지는 못함



(그림 15) 입력 유형과 조건에 따른 규칙 기반 검색 경로 결정

고급 RAG

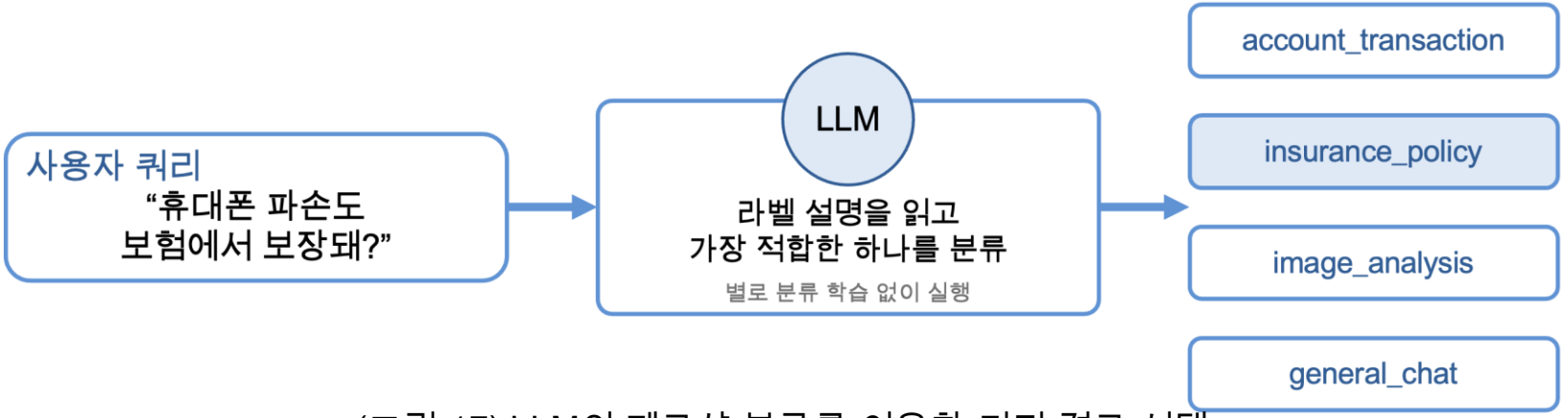
- 쿼리 라우팅 (4/7)
  - 키워드 라우터
    - 개념
      - 쿼리의 키워드와 선택 가능한 경로의 키워드 목록을 매칭하여 처리 경로를 선택함
    - 특징
      - 논리적 라우터보다 조금 더 정교하게 경로를 결정할 수 있음
      - 희소 인코더, 전문 패키지 또는 LLM을 활용하여 구현할 수 있음



(그림 16) 키워드-경로 사전을 이용한 검색 데이터베이스 선택

고급 RAG

- 쿼리 라우팅 (5/7)
  - 제로샷 분류 라우터
    - 개념
      - LLM이 사전에 특정 레이블에 대해 학습되지 않은 상태에서 입력 항목을 주어진 레이블 중 하나로 분류하도록 하는 방식
    - 동작 방식
      1. 쿼리가 입력되면 LLM에 선택 가능한 경로의 레이블 목록을 제공함
      2. LLM이 해당 쿼리에 적합한 경로 레이블을 목록에서 선택함



(그림 17) LLM의 제로샷 분류를 이용한 쿼리 경로 선택

고급 RAG

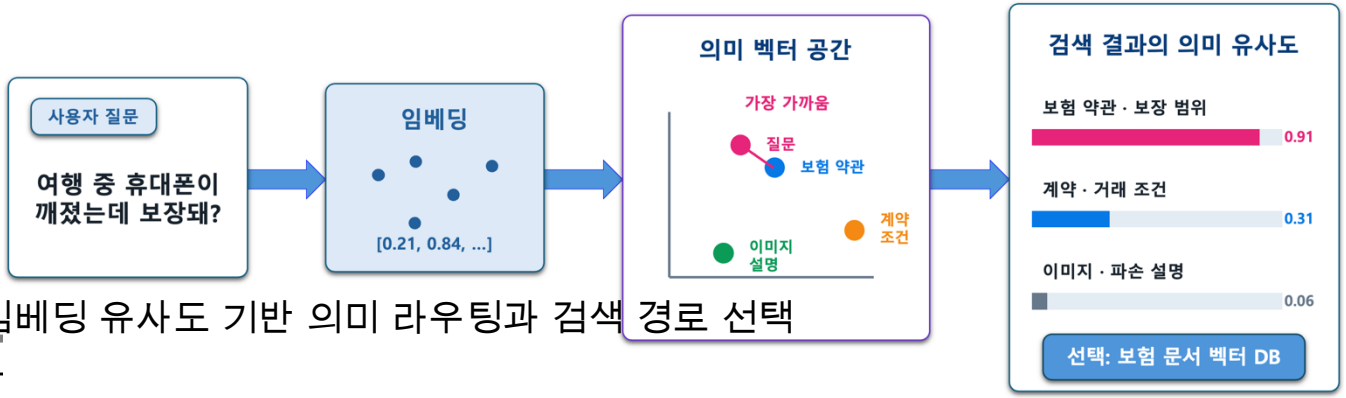
- 쿼리 라우팅 (6/7)
  - LLM 함수 호출 라우터
    - 개념
      - 선택 가능한 여러 경로를 특정 설명과 함께 함수 형태로 정의하고, LLM이 함수 중 하나를 선택하도록 하는 방식
    - 동작 방식
      - LLM의 의사결정 능력을 활용하여 입력된 쿼리를 어떤 함수 또는 경로로 전달할지 결정함

(그림 18) 사용자 쿼리에 따른 함수 호출 기반 검색 경로 결정

고급 RAG

- 쿼리 라우팅 (7/7)
  - 의미 기반 라우터
    - 경로 구성
      - 각 경로를 대표하는 예시 쿼리와 해당 쿼리에 연결된 경로 목록을 준비함
      - 예시 쿼리를 임베딩하여 벡터 데이터베이스에 저장함
    - 경로 선택
      - 새로운 쿼리가 들어오면 저장된 예시 쿼리들과 유사도 검색을 수행함
      - 입력 쿼리와 의미적 유사성이 가장 잘 일치하는 옵션을 선택하여 해당 경로로 전달함

- **보험 문서 경로:** “여행 중 물건이 파손되면 보장받을 수 있나요?”
- **SQL 경로:** “지난달 거래 합계를 알려줘”
- **이미지 분석 경로:** “이 영수증 사진을 읽어줘”



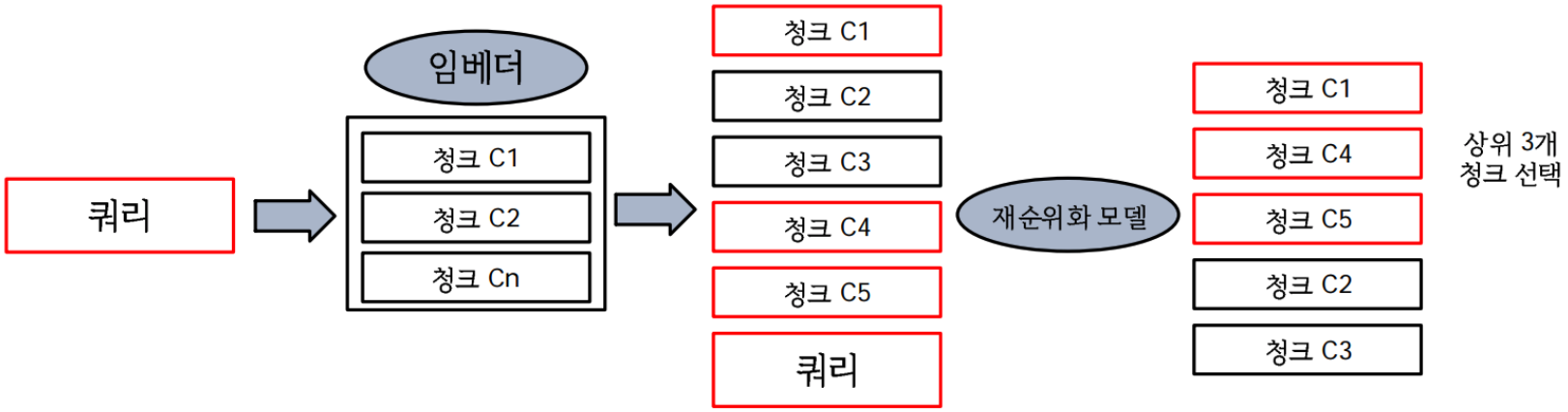
(그림 19) 임베딩 유사도 기반 의미 라우팅과 검색 경로 선택

고급 RAG

- 검색 후 전략 (1/15)
  - 재순위화 (1/8)
    - 개념
      - 검색 단계에서는 가능한 많은 결과를 확보하여 검색 재현율을 높이고, LLM에 제공하는 문서 수는 최소화하여 LLM의 재현율을 극대화하는 방식
    - 처리 과정
      1. 일반적인 검색을 수행하여 많은 수의 청크를 찾음
      2. 재순위화 모델을 이용하여 청크의 순서를 다시 정렬하고, 상위 k개의 청크만 선택하여 LLM에 제공함

고급 RAG

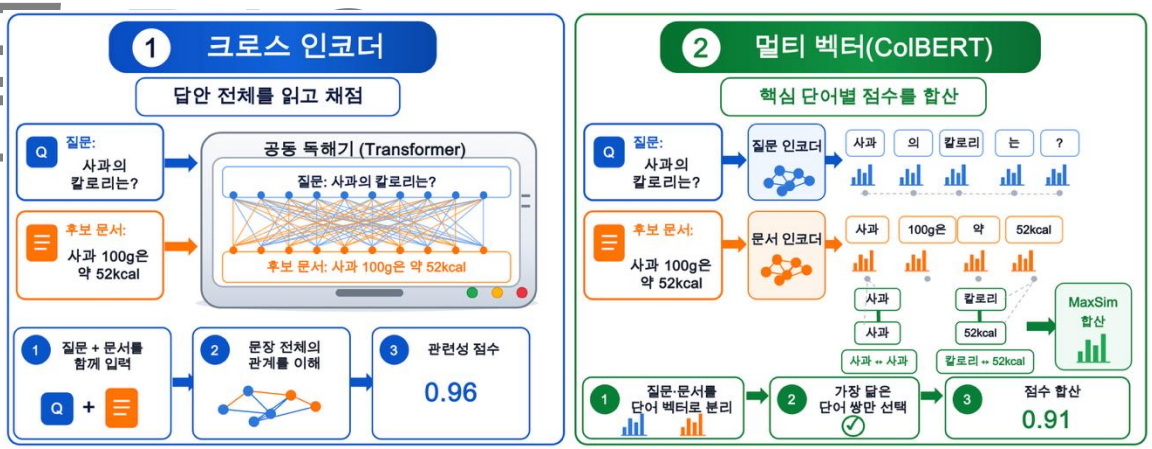
- 검색 후 전략 (2/15)
  - 재순위화 (2/8)
    - 기대 효과
      - LLM에 제공되는 청크의 품질을 높이고 시스템의 할루시네이션을 줄일 수 있음
      - 쿼리와 관련된 상반된 정보까지 고려하여 쿼리의 맥락과 조건에 가장 적합한 청크를 선택함



(그림 20) 임베더 검색 결과의 재순위화와 상위 청크 선택 과정

고급 RAG

- 검색 후 전략 (3/15)
  - 재순위화 (3/8)
    - 크로스 인코더
      - BGE와 같은 트랜스포머 모델에 쿼리와 개별 청크의 두 텍스트 시퀀스를 함께 입력함
      - 쿼리와 청크 사이의 관련성을 0과 1 사이의 유사도 점수로 반환함
    - 멀티 벡터 재순위화 모델
      - 쿼리와 문서를 각각 토큰 수준의 여러 벡터로 표현하고, 두 시퀀스 사이의 정교한 상호작용을 후기 단계에서 수행함
      - 크로스 인코더보다 계산량이 적고, 쿼리와 청크를 입력받아 0과 1 사이의 유사도 점수를 반환함
      - Jina-ColBERT-v1-en과 같이 더 긴 컨텍스트 길이를 지원하는 모델도 있음



(그림 21) 크로스 인코더와 멀티 벡터(ColBERT) 재순위화 과정 비교

고급 RAG

- 검색 후 전략 (4/15)
  - 재순위화 (4/8)
    - LLM 기반 재순위화
      - 포인트와이즈 방식은 쿼리와 하나의 문서 사이의 관련성을 개별적으로 평가함
      - 페어와이즈 방식은 쿼리와 두 문서를 함께 제시하고 두 문서 중 더 관련 있는 문서를 선택함
      - 리스트와이즈 방식은 쿼리와 문서 목록을 제공하고 순위가 매겨진 문서 목록을 출력하도록 지시함
        - 리스트와이즈 방식에는 일반적으로 GPT와 같은 모델이 사용되며 높은 계산 및 경제적 비용이 발생할 수 있음

고급 RAG

- 검색 후 전략 (5/15)
  - 재순위화 (5/8)
    - 파인튜닝된 LLM
      - 일반 LLM은 순위 매기기에 대한 구체적인 훈련을 받지 않았기 때문에 쿼리와 문서의 관련성을 정확하게 측정하지 못할 수 있음
      - 순위 매기기 작업에 특화되도록 파인튜닝하여 재순위화 성능을 향상시킬 수 있음

고급 RAG

- 검색 후 전략 (6/15)
  - 재순위화 (6/8)
    - 성능과 비용
      - 재순위화 방식은 검색 품질뿐만 아니라 연산 비용, 지연 시간 및 시스템 비용에도 영향을 미침
      - 멀티 벡터 모델은 연산 비용이 낮지만 상대적으로 제한된 성능을 보이며, LLM 기반 방식은 성능이 가장 우수하지만 연산 비용이 큼
      - 재순위화는 시스템 전체 성능에 긍정적인 영향을 주기 때문에 RAG 파이프라인의 핵심 구성 요소로 자주 사용됨

Reranker model	Avg.	NQ	HotpotQA	FiQA
***Embedding:* snowflake-arctic-embed-l**	0.6100	0.6311	0.7518	0.4471
+ ms-marco-MiniLM-L-12-v2	0.5771	0.5876	0.7586	0.3850
+ mxbai-rerank-large-v1	0.6077	0.6433	0.7401	0.4396
+ jina-reranker-v2-base-multilingual	0.6481	0.6768	0.8165	0.4511
+ bge-reranker-v2-m3	0.6585	0.6965	0.8458	0.4332
+ NV-RerankQA-Mistral-4B-v3	**0.7529**	**0.7788**	**0.8726**	**0.6073**
***Embedding:* NV-EmbedQA-e5-v5**	0.6083	0.6380	0.7160	0.4710
+ ms-marco-MiniLM-L-12-v2	0.5785	0.5909	0.7458	0.3988
+ mxbai-rerank-large-v1	0.6077	0.6450	0.7279	0.4502
+ jina-reranker-v2-base-multilingual	0.6454	0.6780	0.7996	0.4585
+ bge-reranker-v2-m3	0.6584	0.6974	0.8272	0.4506
+ NV-RerankQA-Mistral-4B-v3	**0.7486**	**0.7785**	**0.8470**	**0.6203**
***Embedding:* NV-EmbedQA-Mistral7B-v2**	0.7173	0.7216	0.8109	0.6194
+ ms-marco-MiniLM-L-12-v2	0.5875	0.5945	0.7641	0.4039
+ mxbai-rerank-large-v1	0.6133	0.6439	0.7436	0.4523
+ jina-reranker-v2-base-multilingual	0.6590	0.6819	0.8262	0.4689
+ bge-reranker-v2-m3	0.6734	0.7028	0.8635	0.4539
+ NV-RerankQA-Mistral-4B-v3	**0.7694**	**0.7830**	**0.8904**	**0.6350**

(그림 22) 임베딩 모델별 재순위화 모델 적용에 따른 질의응답 성능 비교 [1]

[1] G. D. S. P. Moreira, R. Ak, B. Schifferer, M. Xu, R. Osmulski, and E. Oldridge, “Enhancing Q&A text retrieval with ranking models: Benchmarking, fine-tuning and deploying rerankers for RAG,” *arXiv preprint arXiv:2409.07691*, 2024.

고급 RAG

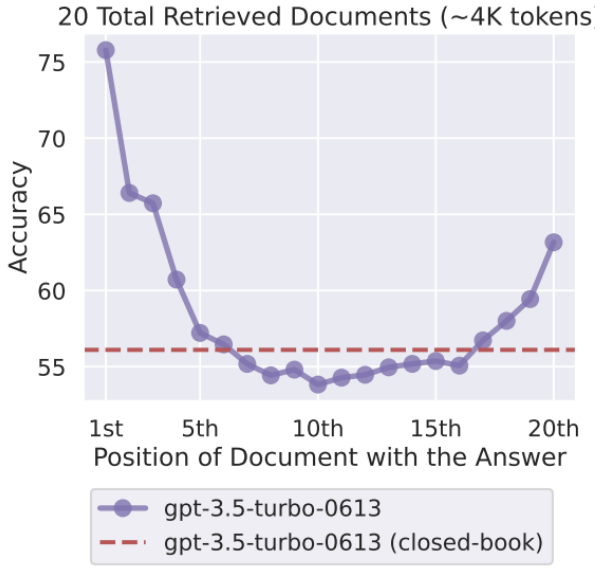
- 검색 후 전략 (7/15)
  - 재순위화 (7/8)
    - 대안적 후처리 기법
      - 청크 필터링
        - 유사도 점수가 특정 임곗값 이하인 청크를 제거할 수 있음
        - 특정 키워드가 포함되지 않았거나 관련 메타데이터에 특정 값이 없는 청크를 제거할 수 있음
        - 특정 날짜 이전에 생성된 청크를 제거하는 등 다양한 기준으로 검색 결과를 필터링할 수 있음

고급 RAG

- 검색 후 전략 (8/15)
  - 재순위화 (8/8)
    - 대안적 후처리 기법
      - k-최근접 이웃 탐색
        - 임베딩 벡터를 이용하여 청크를 찾은 뒤 잠재 공간에서 해당 청크와 이웃 관계에 있는 다른 청크를 추가함
        - 이 과정은 재순위화 전이나 후에 수행할 수 있음
      - 최종 청크 선택
        - LLM에 컨텍스트로 제공할 청크를 최종적으로 선택한 뒤 문서의 배치를 조정할 수 있음
        - 중요한 정보가 입력 컨텍스트의 앞부분이나 뒷부분에 배치될 때 LLM의 성능이 가장 높음
        - 컨텍스트가 매우 길면 중간에 배치된 정보의 성능이 낮아질 수 있음

고급 RAG

- 검색 후 전략 (9/15)
  - 청크 재배치
    - 배치 방식
      - 청크를 관련성 순서대로 배치하거나 교차 패턴을 이용하여 재배치할 수 있음
      - 교차 패턴에서는 짝수 인덱스의 청크를 목록 앞부분에, 홀수 인덱스의 청크를 뒷부분에 배치함
    - 적용 목적
      - 관련성이 가장 높은 청크를 입력 컨텍스트의 앞과 뒤에 배치함
      - 상대적으로 중요성이 낮은 청크는 입력 컨텍스트의 중간에 배치함
      - 넓은 범위의 상위 k개 청크를 사용할 때 효과적이며 시스템 성능을 개선할 수 있음



(그림 23) 검색 문서 내 정답 정보의 위치에 따른 LLM 정확도 변화 [2]

[2] N. F. Liu *et al*, "Lost in the middle: How language models use long contexts," *Trans. Assoc. Comput. Linguistics*, vol. 12, pp. 157–173, 2024.

고급 RAG

- 검색 후 전략 (10/15)
  - 출처 인용
    - 필요성
      - 시스템 성능 자체를 개선하는 기법은 아니지만 RAG 시스템의 구성 요소로 강력히 권장됨
      - 여러 출처를 이용하여 응답을 생성할 때 어떤 문서가 사용되었는지 추적하는 것이 중요함
    - 적용 방식
      - 응답 생성에 사용된 문서 또는 청크가 속한 원문을 기록할 수 있으며, LLM의 프롬프트에 사용된 출처를 명시할 수 있음
      - 퍼지 인용 질의 엔진을 사용하여 생성된 응답과 검색된 청크를 문자열 기반으로 매칭할 수 있음
      - 퍼지 매칭은 청크의 단어를 n-그램으로 분할한 뒤 TF-IDF를 적용하는 절차에 기반함

고급 RAG

- 검색 후 전략 (11/15)
  - 챗엔진
    - 개념
      - RAG를 이용하여 LLM이 사용자와 나눈 과거 대화를 기억하도록 하는 확장 기법
    - 적용 방식
      - 이전 대화를 프롬프트에 포함하는 단순한 방식을 사용할 수 있으나, 대화가 길어지면 토큰 수와 비용이 증가하고, 오래된 무관한 대화까지 포함 가능함
      - 대화를 임베딩하여 핵심 부분만 추출하거나 사용자 대화의 흐름을 포착할 수 있음
      - 여러 메시지에 걸쳐 대화가 이어질 경우 이전 대화 내용을 요약하여 프롬프트 길이를 줄이는 프롬프트 압축을 수행할 수 있음

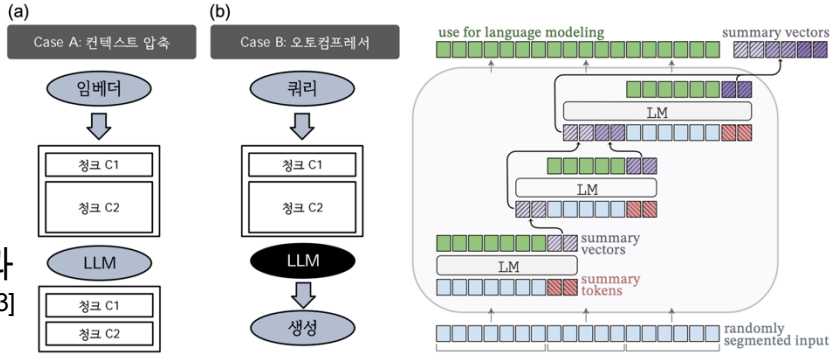
- "세 가지 보험 상품을 비교해줘"
- "그 중 자기부담금이 가장 낮은 것은?"

고급 RAG

- 검색 후 전략 (12/15)
  - 컨텍스트 압축 (1/2)
    - 개념
      - 문서 검색 후 컨텍스트를 압축하여 관련 정보만 보존하고, LLM의 응답 생성을 지원하면서 연산 자원이나 API 사용 비용을 줄이는 방식
    - 종류
      - 컨텍스트 필터링
        - 정보 이론에 따라 엔트로피가 낮은 토큰은 예측하기 쉽고 중복 정보를 포함하므로 관련성이 낮고 맥락 이해에 거의 도움이 되지 않음
        - 각 어휘 단위에 정보 값을 할당한 뒤 내림차순으로 순위를 매김
        - 사전에 정하거나 컨텍스트에 따라 설정한 p번째 백분위수 이내의 토큰만 유지함

고급 RAG

- 검색 후 전략 (13/15)
  - 컨텍스트 압축 (2/2)
    - 종류



(그림 24) 컨텍스트 필터링·압축 방식과 오토컴프레서의 요약 벡터 생성 구조 [3]

      - LongLLMLingua
        - 정보 엔트로피를 기반으로 하며 컨텍스트와 쿼리의 정보를 모두 사용함
        - 문서를 동적으로 압축하고 재배치하여 LLM의 응답 생성을 더 효율적으로 만듦
      - 오토컴프레서
        - 시스템의 파인튜닝과 요약 벡터를 활용하여 긴 텍스트를 작은 벡터 표현으로 압축하며, 생성된 요약 벡터를 소프트 프롬프트로 사용하여 모델에 컨텍스트를 제공함
        - LLM의 가중치는 고정된 상태로 유지하고 프롬프트에 삽입되는 학습 가능한 토큰만 훈련하며, 학습 가능한 토큰을 최적화하여 모델의 핵심 파라미터를 수정하지 않고 시스템을 엔드 투 엔드 방식으로 최적화함

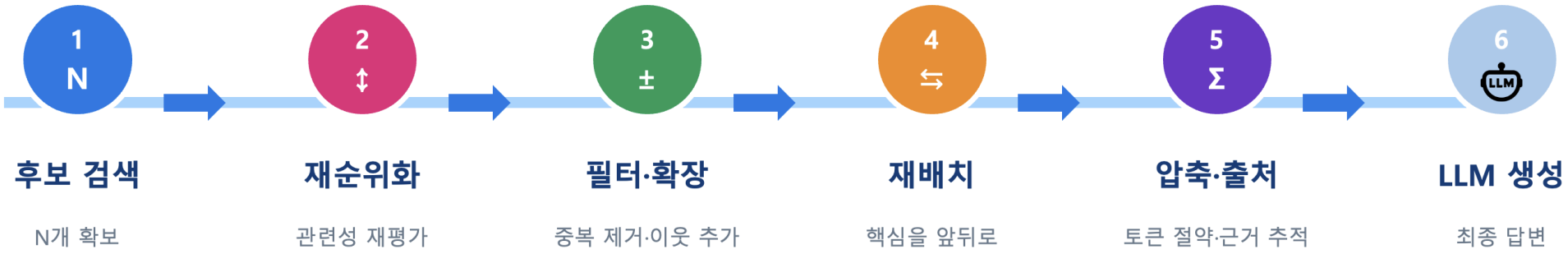
[3] A. Chevalier, A. Wettig, A. Ajith, and D. Chen, “Adapting language models to compress contexts,” in *Proc. 2023 Conf. Empirical Methods Natural Language Process. (EMNLP)*, pp. 3829–3846, 2023.

고급 RAG

- 검색 후 전략 (14/15)
  - 프롬프트 엔지니어링
    - 적용 목적
      - LLM과의 모든 상호작용에 공통으로 적용하여 생성 품질을 향상시킴
      - RAG 시스템에 가장 적합한 프롬프트를 설계하기 위해 구체적인 지침과 예시를 활용함
    - 적용 방식
      - “컨텍스트를 사용해 답하시오”와 같이 명확한 지시를 제공함
      - “컨텍스트에 답이 없으면 모른다고 하시오”와 같이 답변 조건을 설정함
      - 목록이나 HTML 등 출력 형식을 특정 방식으로 지정할 수 있음
      - RAG용 프롬프트 작성을 지원하는 라이브러리를 활용할 수도 있음

고급 RAG

- 검색 후 전략 (15/15)
  - 전체 흐름



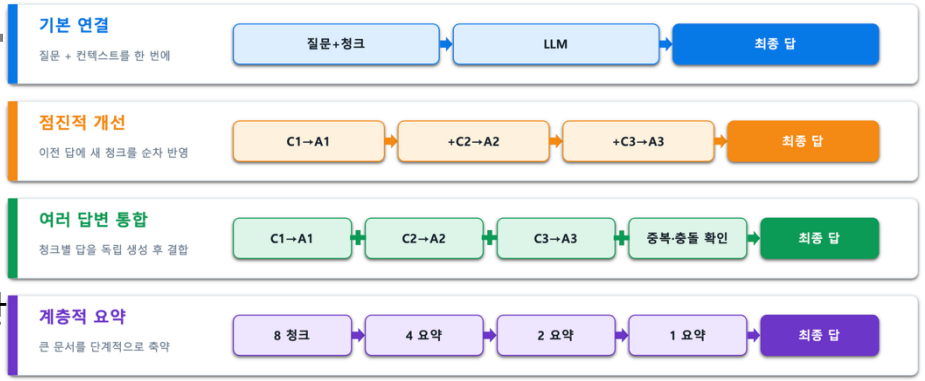
(그림 25) 고급 RAG의 검색 후 처리 전체 흐름

고급 RAG

- 응답 최적화 (1/3)
  - 개념
    - RAG 파이프라인의 마지막 단계로, 검색된 정보를 바탕으로 생성한 최종 응답을 사용자에게 제공하기 전에 개선하는 과정
  - 응답 합성기
    - 사용자 쿼리·검색된 컨텍스트·프롬프트를 함께 LLM에 전달하여 응답을 생성함
    - 기본 방식은 LLM을 한 번 호출하지만, 보다 정교한 응답을 위해 LLM을 여러 번 호출할 수 있음

고급 RAG

- 응답 최적화 (2/3)
  - 응답 합성 방식
    - 점진적 개선
      - 한 번에 하나의 청크를 사용하여 응답을 반복적으로 개선함
      - 이전 응답과 다음 청크를 함께 입력하여 새로운 정보가 반영되도록 응답을 보완함
    - 다단계 응답 합성
      - 서로 다른 청크를 이용해 여러 응답을 생성함
      - 생성된 응답을 모두 연결한 뒤 최종 요약 응답을 생성함
    - 계층적 요약
      - 서로 다른 컨텍스트에서 생성된 응답을 시작점으로 삼아 재귀적으로 결합함
      - 요약과 생성된 답변의 품질을 높일 수 있지만, LLM 호출 증가로 연산 자원과 비용 부담이 커짐



(그림 26) 검색 컨텍스트를 활용한 네 가지 응답 합성 방식 비교

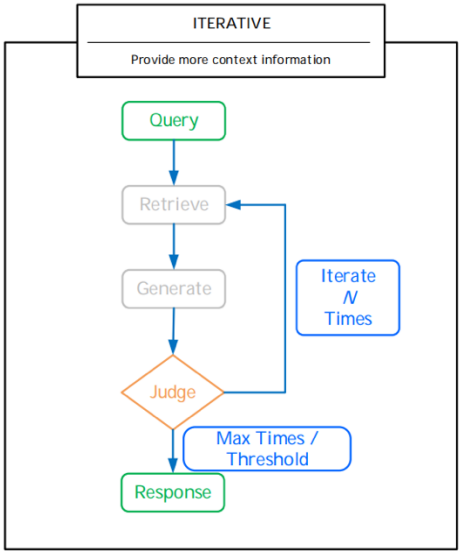
고급 RAG

- 응답 최적화 (3/3)
  - RAG와 에이전트의 결합
    - RAG는 에이전트 시스템에서 시스템의 메모리 역할을 수행할 수 있음
    - 에이전트는 대화 이력 검색, 쿼리 라우팅, API 연결, 코드 실행 같은 다양한 구성 요소를 다룰 수 있음
    - LLM은 사용자의 주문을 수행하면서 RAG·도구 호출·웹사이트 연결 등의 추가 작업을 수행할 수 있음

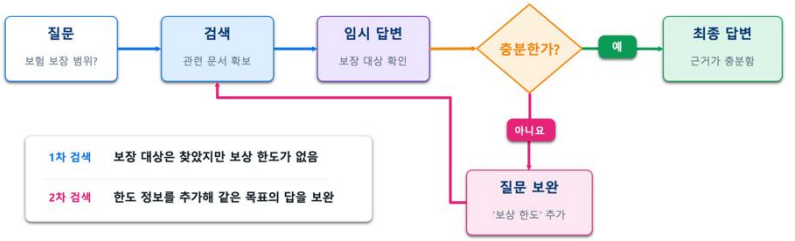
(그림 27) 사용자 의도에 따른 에이전트의 응답 경로 선택과 결과 통합

고급 RAG

- RAG 파이프라인 확장 (1/3)
  - 다단계 추론 (1/3)
    - 개념
      - 검색을 한 번 수행하고 곧바로 응답을 생성하는 방식은 복잡한 문제를 해결하는 데 충분하지 않을 수 있음에 따라 검색과 판단을 반복하는 방식
    - 종류
      - 반복 검색
        - 동작 방식
          - 하나의 쿼리에 대해 검색과 응답 생성을 수행한 뒤 LLM이 결과를 평가를 수행하며, 평가 결과에 따라 최대 n번까지 검색·생성·평가 과정을 반복함
        - 효과와 한계
          - 반복할수록 응답의 견고성이 향상될 수 있음
          - 동시에 불필요한 정보가 축적될 위험이 있음



(그림 28) 반복 검색·생성·평가를 통한 반복형 RAG의 동작 구조 [4]

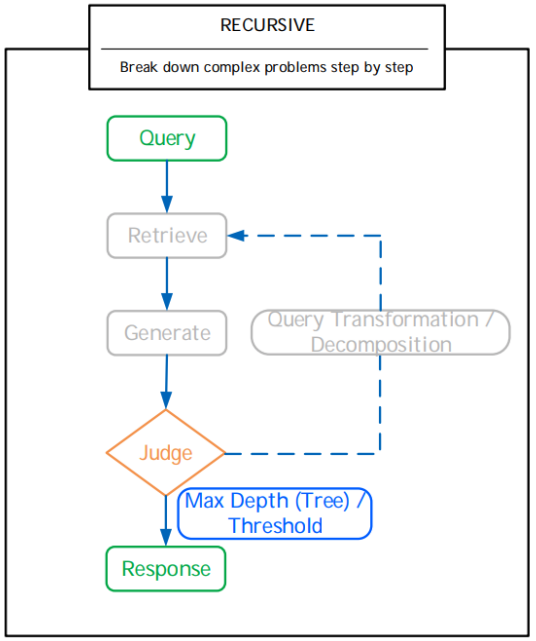


(그림 29) 답변 충분성 평가와 질문 보완을 이용한 반복 검색 과정

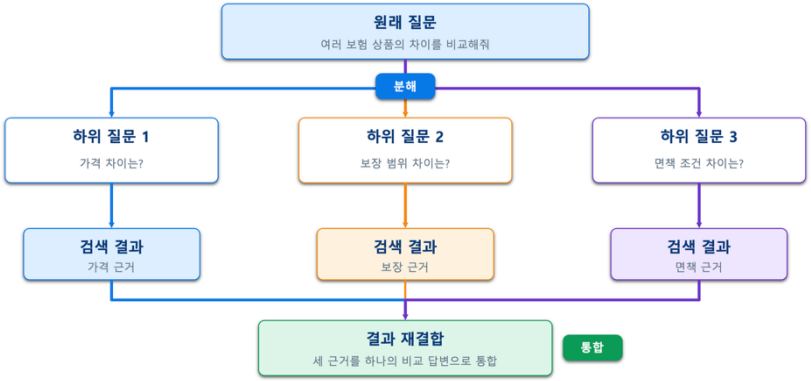
[4] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

고급 RAG

- RAG 파이프라인 확장 (2/3)
  - 다단계 추론 (2/3)
    - 종류
      - 재귀 검색
        - 동작 방식
          - 이전 검색 결과를 바탕으로 쿼리를 점진적으로 개선하며 검색의 깊이와 관련성을 높이며, 검색 결과와 쿼리 개선 사이에 피드백 루프가 형성됨
        - 활용
          - 사고의 사슬 기법을 통해 쿼리를 중간 단계로 분해하고 순차적으로 해결함
          - 쿼리가 명확하지 않거나 전문적·세부적인 정보를 신중하게 탐색해야 할 때 유용함



(그림 30) 쿼리 분해·변환을 반복하는 재귀형 RAG의 동작 구조 [4]

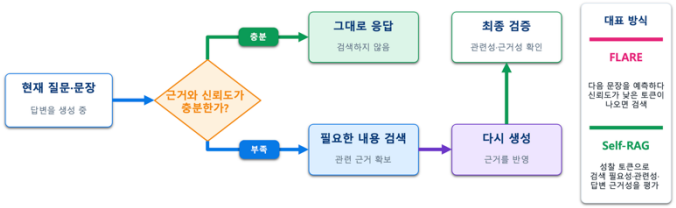


(그림 31) 복합 질문의 하위 질문 분해와 검색 결과 재결합 과정

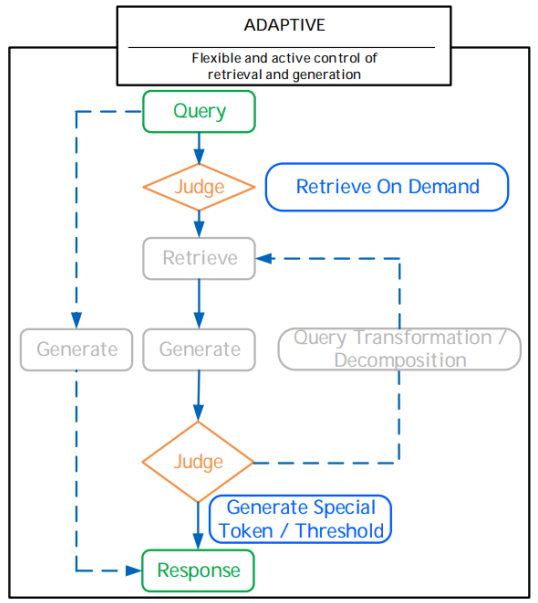
[4] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

고급 RAG

- RAG 파이프라인 확장 (3/3)
  - 다단계 추론 (3/3)
    - 종류
      - 적응형 검색
        - 동작 방식
          - LLM이 검색 시점과 검색된 콘텐츠의 적절성을 스스로 판단함
          - 언제 응답할지, 언제 검색할지, 추가적인 도움이 필요한지를 결정함
        - 활용
          - Flare
            - 모델이 다음에 작성할 문장을 미리 예측하고, 그 문장에 신뢰도가 낮은 토큰이 포함되면 해당 내용을 검색어로 사용함
            - 신뢰도가 일정 임계치 이하로 떨어지면 추가 검색 여부를 결정함
          - Self-RAG
            - 모델이 검색 필요성, 검색 문서의 관련성, 생성된 답변이 근거로 뒷받침되는지를 평가하도록 성찰 토큰을 학습시킴
            - 이를 통해 필요할 때 검색하고 검색 결과와 자신의 답변을 함께 비판적으로 평가하도록 함



(그림 33) 근거 충분성 판단을 이용한 적응형 RAG의 검색·재생성 과정



(그림 32) 검색 필요성과 생성 품질을 동적으로 판단하는 적응형 RAG의 동작 구조[4]

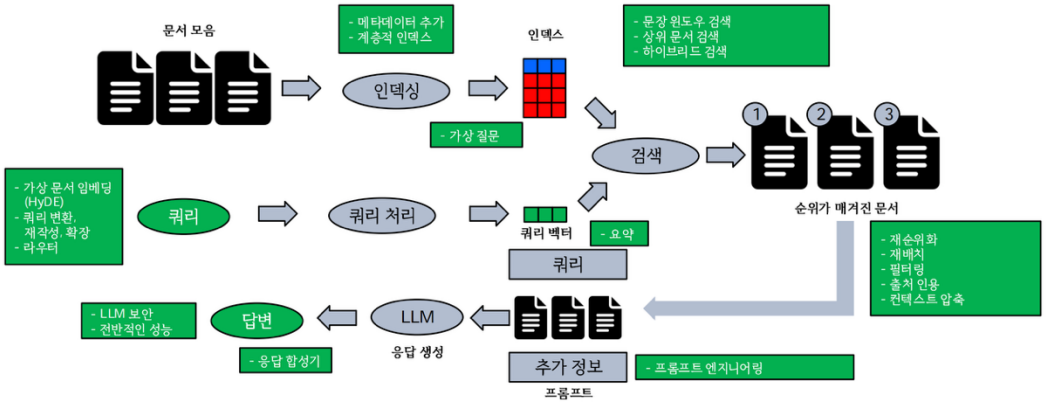
[4] Y. Gao *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.

고급 RAG

• 고급 RAG의 문제 해결 (1/3)

[표 2] RAG 파이프라인의 주요 문제와 고급 RAG 기법 기반 해결 방안

문제	해결책
나이브 RAG의 문제점: 대규모 혹은 장문의 문서로 인한 지연 및 성능 저하	• **계층적 인덱싱 활용:** 큰 단위의 문서를 요약하고 multi-level embedding을 생성하며 메타데이터를 활용함 • 긴 문서에는 맵-리듀스 방식, 다양한 주제를 포함하는 경우에는 다중 요약과 같은 변형 기법을 적용함
코퍼스에 고유한 구조가 있을 때, 평면적 계층 구조가 검색의 관련성을 제한하는 문제	• **계층적 인덱싱 적용:** 문서의 구조(장, 절, 소절)를 존중하고 계층적 요약과 임베딩을 기반으로 컨텍스트를 검색함
낮은 검색 정확도와 특정 도메인을 일반화하는 데 따르는 어려움	• **가상 질문과 HyDE 적용:** 각 청크에 대해 가상 질문을 생성하고 임베딩을 수행함 • 쿼리 의미와 일치하는 가상 답변을 생성하고 임베딩하여 관련 청크를 검색함
세분화된 청크 분할 시 컨텍스트 손실	• **컨텍스트 강화 활용:** 문장 윈도우를 사용하여 검색된 청크를 주변 컨텍스트로 확장하거나 상위 문서를 검색하여 컨텍스트를 확장함
복잡한 쿼리 및 초기 검색에서 낮은 재현율	• 쿼리 변환 적용: 복잡한 쿼리를 하위 쿼리로 분해하고, 스텝백 프롬프팅 또는 쿼리 확장을 활용함 • 변환된 쿼리를 임베딩하여 검색 성능을 향상함



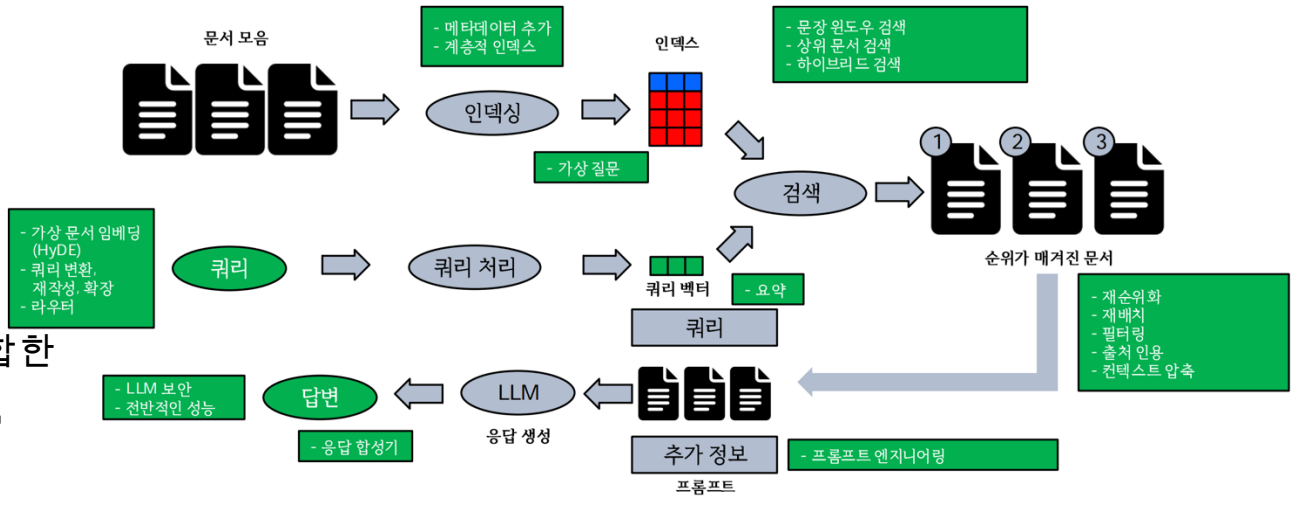
(그림 34) 인덱싱·쿼리 처리·검색 후 전략을 통합한 고급 RAG 파이프라인

고급 RAG

• 고급 RAG의 문제 해결 (2/3)

[표 2] RAG 파이프라인의 주요 문제와 고급 RAG 기법 기반 해결 방안

문제	해결책
특정 용어나 키워드에 대한 컨텍스트 불일치	• **하이브리드 검색 활용:** 키워드 기반(e.g., BM25)과 벡터 기반 검색을 가중치 점수를 사용하여 결합함
다양한 유형의 쿼리를 비효율적으로 처리하는 문제	• **쿼리 라우팅 구현:** 논리적 규칙, 키워드 기반 또는 시멘틱 분류기, 제로샷 모델, LLM 기반 라우터를 사용하여 쿼리를 적절한 벡엔드로 전달함
임의의 상위 k개 차단(cut-off)으로 인한 관련 청크 손실	• **재순위와 적용:** 크로스 인코더, 멀티 벡터 재순위화 모델 또는 LLM 기반(포인트와이즈, 페어와이즈, 리스트와이즈) 재순위화를 사용하여 검색된 청크를 재배치함
LLM 컨텍스트 내 정보 손실 또는 비효율성	• **컨텍스트 압축 적용:** 엔트로피가 낮은 토큰을 필터링하거나 청크를 동적으로 압축하고 재배치(LongLLMLingua), 요약 벡터 및 오토컴프레서 기법을 활용함
비효율적인 응답 생성	• **응답 최적화:** 반복적 개선, 계층적 요약, 다단계 응답 합성을 활용하며, 프롬프트 품질과 구체성을 개선함



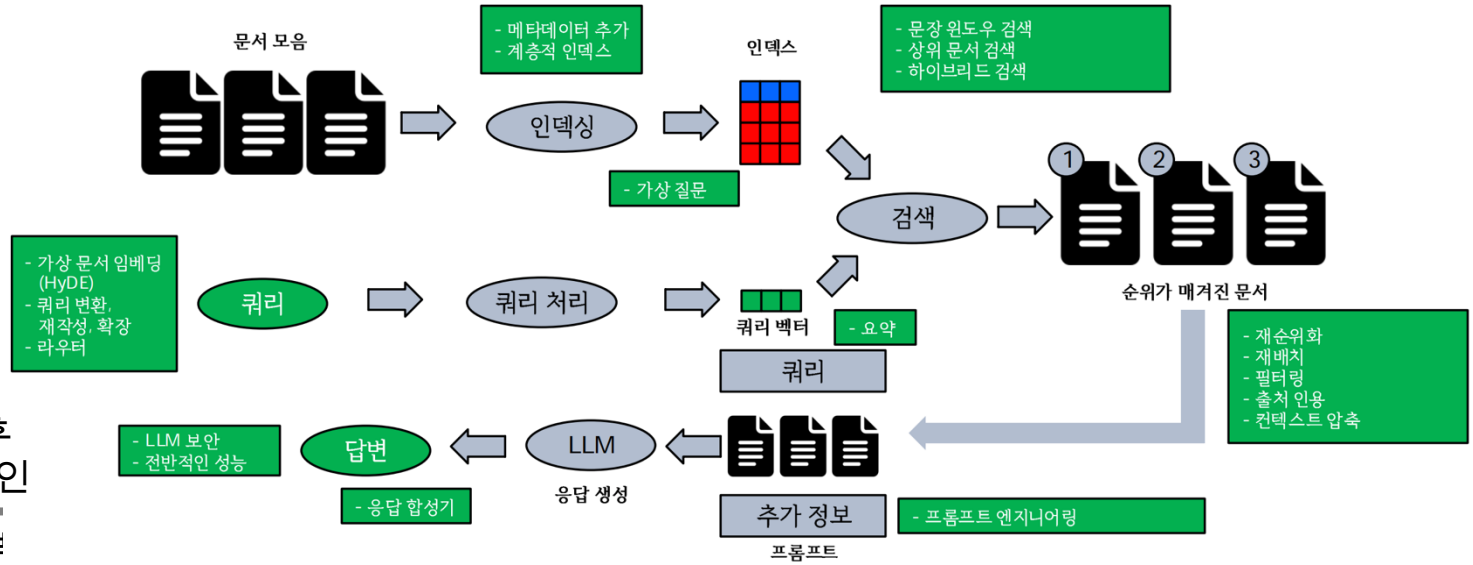
(그림 34) 인덱싱·쿼리 처리·검색 후 전략을 통합한 고급 RAG 파이프라인

고급 RAG

- 고급 RAG의 문제 해결 (3/3)

[표 2] RAG 파이프라인의 주요 문제와 고급 RAG 기법 기반 해결 방안

문제	해결책
대화형 시스템에서의 메모리 한계	• **챗엔진(chatengine) 기법 활용**: 과거 대화를 저장하고 임베딩하며, 사용자 대화를 압축하고 채팅 이력을 현재 쿼리와 병합함
복잡한 추론 또는 동적 쿼리 적응의 필요성	• **적응형 및 다단계 검색 채택:** 피드백 루프와 자기 성찰(e.g., Flare, Self-RAG)을 사용하여 재귀적, 반복적, 적응형 검색의 접근 방식을 사용함
생성된 응답에서 출처 추적 부재	• **인용 포함:** 퍼지 인용 매칭, 메타데이터 태그를 사용하거나 프롬프트에 출처 참조를 포함함
쿼리 복잡성 또는 모달리티를 기반으로 한 파이프라인 맞춤화 필요성	• **RAG 파이프라인 확장:** 에이전트와 결합해 추론, 도구 사용, 의사결정을 수행함 • 복잡한 쿼리에 대해서는 적응형 및 재귀적 검색 루프를 적용함

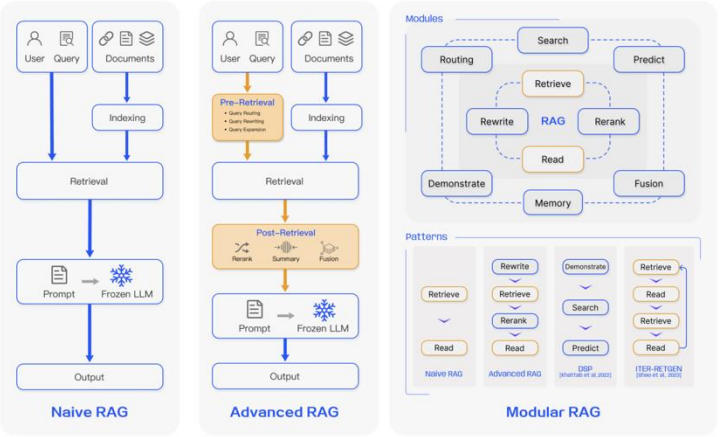


(그림 34) 인덱싱·쿼리 처리·검색 후 전략을 통합한 고급 RAG 파이프라인

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

모듈형 RAG 기반 시



(그림 35) RAG의 세 가지 주요 패러다임[4]

- 모듈형 RAG (1/6)
  - 개념
    - 고급 RAG의 확장 개념으로, 특히 시스템의 적응성과 확장성에 초점을 맞춘 발전된 형태로, 시스템을 개별 컴포넌트인 모듈로 나누고, 이를 순차적 또는 병렬로 활용함
  - 파이프라인 구조
    - 검색과 생성이 서로 번갈아 수행되는 형태로 파이프라인을 재구성함
    - 다양한 작업에 유연하게 대응할 수 있도록 기능별로 특화된 모듈을 도입함
  - 목표
    - RAG 시스템의 성능을 최적화함
    - 작업에 따라 필요한 모듈과 실행 흐름을 유연하게 조정함

[4] Y. Gao *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023.

모듈형 RAG 기반 시스템 통합

- 모듈형 RAG (2/6)
  - 주요 모듈 (1/2)
    - 검색 모듈
      - 사용자 쿼리와 관련된 정보를 탐색함
      - 검색 엔진·데이터베이스·지식 그래프 및 정교한 검색 알고리즘·머신러닝 기법·코드 실행까지 활용할 수 있음
    - 메모리 모듈
      - 검색 과정에서 확인한 관련 정보를 저장함
      - 이전에 검색한 컨텍스트를 다시 탐색하여 가져올 수 있음
    - 라우팅 모듈
      - 쿼리에 가장 적합한 처리 경로를 식별함
      - 서로 다른 데이터베이스에서 각각 필요한 정보를 검색하거나 복잡한 쿼리를 여러 개로 분해할 수 있음

모듈형 RAG 기반 시스템 통합

- 모듈형 RAG (3/6)
  - 주요 모듈 (2/2)
    - 생성 모듈
      - 쿼리에 따라 요약·의역·맥락 확장 등 서로 다른 유형의 응답을 생성함
      - 생성 결과의 품질과 관련성을 높이는 데 초점을 맞춤
    - 작업 적응 모듈
      - 요청된 작업에 동적으로 적응하도록 시스템의 동작을 조정함
      - 작업에 따라 검색·처리·생성 과정을 유연하게 변경함
    - 검증 모듈
      - 검색된 응답과 컨텍스트를 평가하며, 오류·편향·정보 간 불일치를 식별함
      - 평가와 검증을 반복하여 시스템의 응답을 점차 개선함

모듈형 RAG 기반 시스템 통합

- 모듈형 RAG (4/6)
  - 장점
    - 높은 적응성
      - 각 모듈을 필요에 따라 교체하거나 재구성할 수 있음
      - 모듈 사이의 흐름을 세밀하게 조정하여 추가적인 유연성을 확보함
    - 파이프라인 확장
      - 나이브 RAG와 고급 RAG가 주로 '검색 및 읽기'에 초점을 맞춤
      - 모듈형 RAG는 '검색 → 읽기 → 다시 쓰기'가 가능한 구조로 확장됨
    - 응답 개선
      - 평가 및 피드백 기능을 통해 쿼리에 대한 응답을 정교하게 다듬을 수 있음
      - 검색과 생성 사이에 다양한 모듈을 결합하여 작업별로 다른 실행 흐름을 구성할 수 있음

모듈형 RAG 기반 시스템 통

- 모듈형 RAG (5/6)
  - 활용 (1/2)
    - 파라메트릭 메모리 기반 방식
      - LLM의 파라메트릭 메모리에 저장된 정보와 검색 결과를 통합함
      - 모델이 검색을 수행하기 전에 먼저 응답을 생성하도록 요청하는 '암송하고 답하기' 방식을 사용할 수 있음
    - DSP (Demonstrate-Search-Predict) 접근법
      - 시연-검색-예측의 순서로 LLM과 RAG를 결합함
      - 복잡한 쿼리 또는 지식 집약적 작업을 해결하기 위한 다양한 상호작용 방식을 제공함
      - 견고하면서도 유연한 모듈형 RAG 파이프라인을 구현할 수 있음을 보여줌

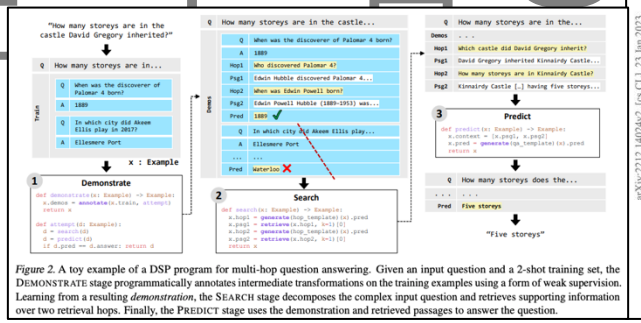
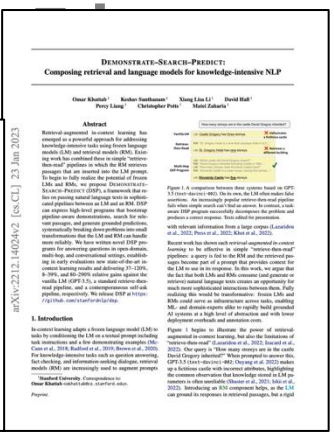


Figure 2. A toy example of a DSP program for multi-hop question answering. Given an input question and a 2-shot training set, the DEMONSTRATE stage programmatically annotates intermediate transformations on the training examples using a form of weak supervision. Learning from a resulting *demonstration*, the SEARCH stage decomposes the complex input question and retrieves supporting information over two retrieval hops. Finally, the PREDICT stage uses the demonstration and retrieved passages to answer the question.


DEMONSTRATE–SEARCH–PREDICT:
Composing retrieval and language models for knowledge-intensive NLP

(그림 36) DSP 프레임워크의 시연·검색·예측 기반 다중 홉 질의응답 구조[5]

[5] O. Khattab *et al.*, "Demonstrate-search-predict: Composing retrieval and language models for knowledge-intensive NLP," *arXiv preprint arXiv:2212.14024*, 2022.

모듈형 RAG 기반 시스템 통합

- 모듈형 RAG (6/6)
  - 활용 (2/2)
    - Self-RAG
      - 시스템에 비판적이고 자기 성찰적인 요소를 도입함
      - LLM이 자신이 생성한 내용을 반성하고 사실성과 전반적인 품질 측면에서 결과를 비판함

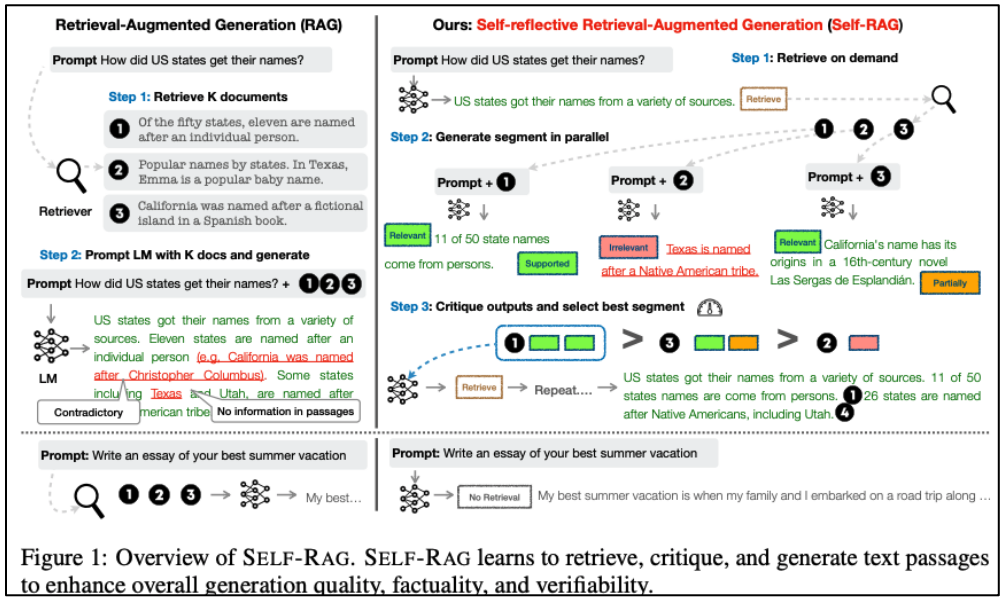


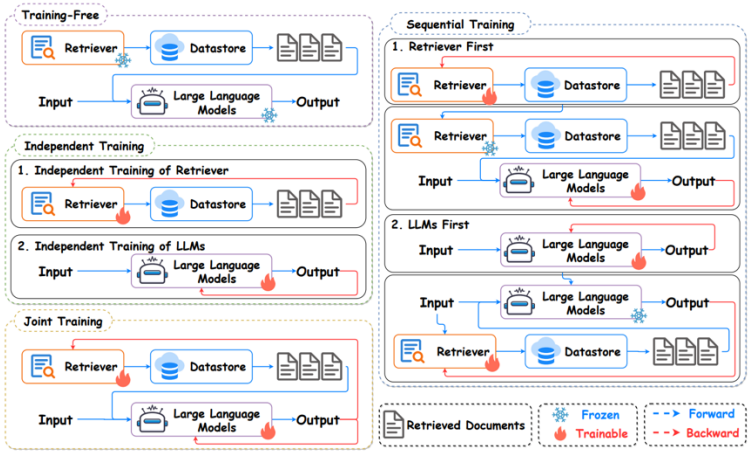
Figure 1: Overview of SELF-RAG. SELF-RAG learns to retrieve, critique, and generate text passages to enhance overall generation quality, factuality, and verifiability.

(그림 37) 검색·생성·비평을 자기 성찰적으로 수행하는 Self-RAG 구조 [6]

[6] A. Asai, Z. Wu, Y. Wang, A. Sil, and H. Hajishirzi, "Self-RAG: Learning to retrieve, generate, and critique through self-reflection," in *Proc. Int. Conf. Learning Representations (ICLR)*, pp. 9112–9141, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법과 비훈련 접근법
  - 접근법 구분
    - 비훈련 접근법
      - 임베더와 LLM을 처음부터 고정된 상태로 유지함
      - 두 구성 요소가 이미 사전 학습되어 있으므로 별도의 훈련 없이 활용할 수 있음
      - 나이브 RAG는 일반적으로 비훈련 접근법으로 간주함
    - 훈련 기반 접근법
      - 검색기와 LLM을 목적에 맞게 추가로 훈련함
      - 독립 훈련·순차 훈련·결합 훈련으로 구분할 수 있음

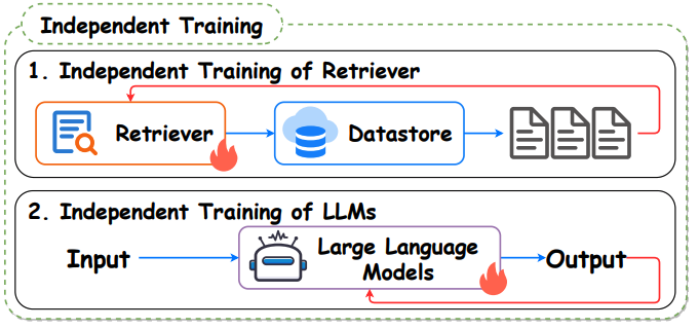


(그림 38) RAG의 다양한 훈련 방법 [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (1/9)
  - 독립 훈련
    - 동작 방식
      - 검색기와 LLM을 완전히 분리된 과정에서 각각 훈련함
      - 훈련 과정에서 두 구성 요소 사이의 상호작용이 발생하지 않음
      - 시스템의 여러 구성 요소를 개별적으로 파인튜닝함
    - 활용 분야
      - 법률·금융·의료 등 특정 도메인에 시스템을 적합하게 조정할 때 유용함
    - 기대 효과
      - 비훈련 접근법보다 애플리케이션 도메인에 대한 시스템 성능을 높일 수 있음
      - LLM이 검색된 컨텍스트를 더욱 효과적으로 활용하도록 파인튜닝할 수 있음

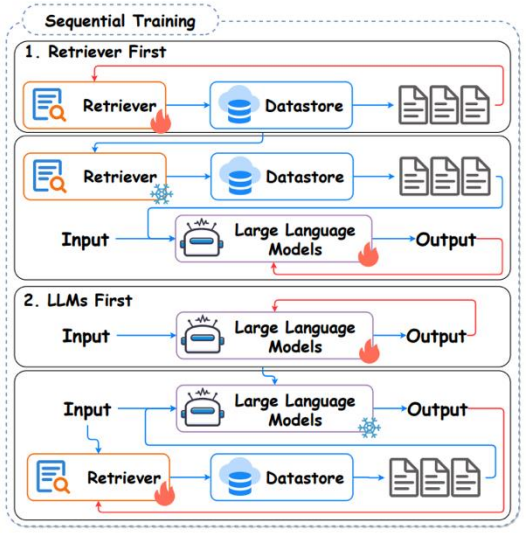


(그림 39) RAG 구성 요소의 독립 훈련 방식 [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템

- 훈련 기반 접근법 (2/9)
  - 순차 훈련 (1/2)
    - 동작 방식
      - 검색기와 LLM을 먼저 독립적으로 훈련한 뒤 순차적으로 추가 훈련함
      - 하나의 구성 요소를 고정하고 다른 구성 요소만 훈련함
      - 훈련 순서에 따라 검색기 우선과 LLM 우선 방식으로 구분함
    - 검색기 우선
      - 검색기를 먼저 훈련한 뒤 고정함
      - 검색된 청크를 이용해 LLM을 파인튜닝함
      - LLM은 검색기의 청크를 받아들이고 컨텍스트 생성에 효과적으로 활용하는 방법을 학습함

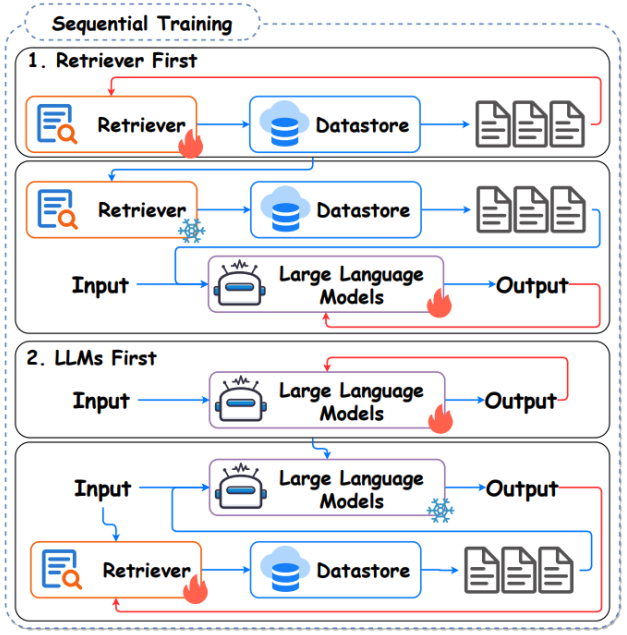


(그림 40) RAG 구성 요소의 순차 훈련 방식 [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (3/9)
  - 순차 훈련 (2/2)
    - LLM 우선
      - LLM의 감독을 활용하여 검색기를 훈련함
      - 많은 파라미터와 학습 데이터를 보유한 LLM이 검색기 훈련의 감독자 역할을 수행함
      - 큰 모델의 지식을 작은 모델에 전달하는 지식 증류의 한 형태로 볼 수 있음

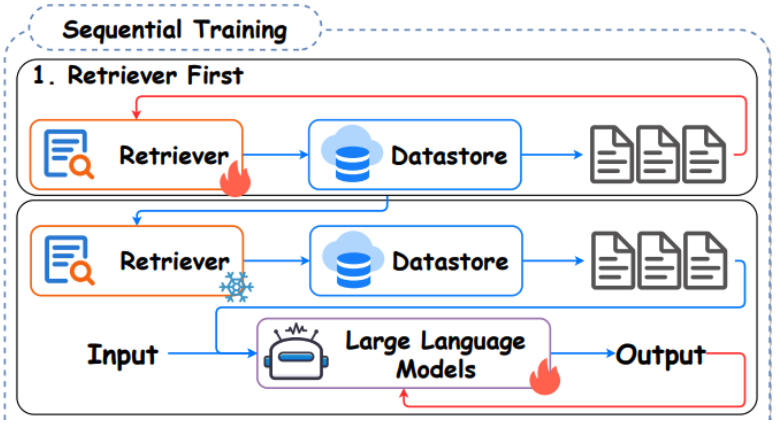


(그림 40) RAG 구성 요소의 순차 훈련 방식 [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (4/9)
  - 검색기 우선 적용 분야 (1/2)
    - 검색 엔진
      - 법률 문서·의료 논문·전자상거래 상품 검색
      - 방대한 정형·비정형 코퍼스에서 관련성 높은 문서를 빠르고 정확하게 검색해야 함
    - 기업 지식 관리 시스템
      - 내부 문서·FAQ 시스템
      - 생성 품질보다 독점 데이터베이스에서 정확한 문서를 찾는 검색 품질이 중요함

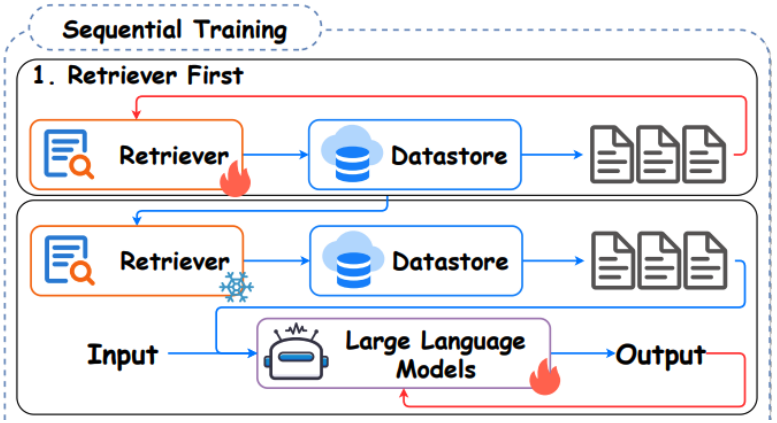


(그림 41) RAG 구성 요소의 순차 훈련 방식 (검색기 우선) [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (5/9)
  - 검색기 우선 적용 분야 (2/2)
    - 과학 연구 저장소
      - PubMed·arXiv·특허 데이터베이스
      - 전문 분야의 검색 정확도와 재현율을 최적화해야 함
    - 규제 및 규정 준수 시스템
      - 금융 규제 준수 점검·판례 데이터베이스
      - 관련성 높은 콘텐츠를 안정적으로 찾고 신뢰도가 낮은 결과를 최소화해야 함

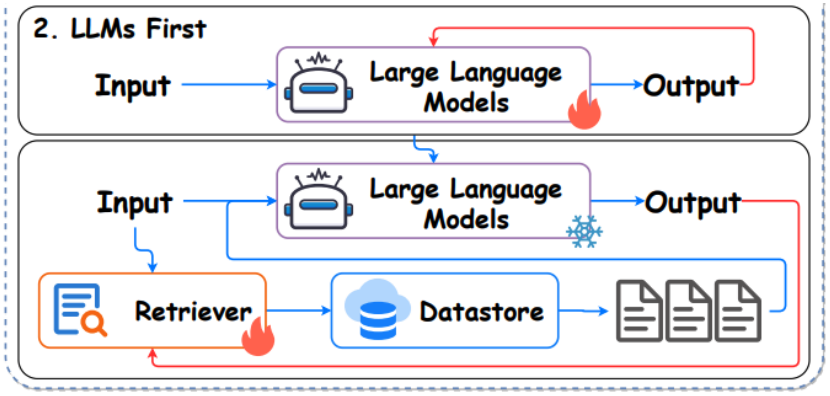


(그림 41) RAG 구성 요소의 순차 훈련 방식 (검색기 우선) [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (6/9)
  - LLM 우선 적용 분야 (1/2)
    - 대화형 에이전트
      - 고객 지원 챗봇·개인 비서
      - LLM이 검색된 콘텐츠를 해석·통합하여 자연스럽고 맥락을 반영한 응답을 생성함
    - 창의적 응용
      - 콘텐츠 작성·스토리텔링·브레인스토밍
      - LLM의 창작·통합·추론 능력이 중심이며, 검색은 배경 지식을 제공하는 역할을 함

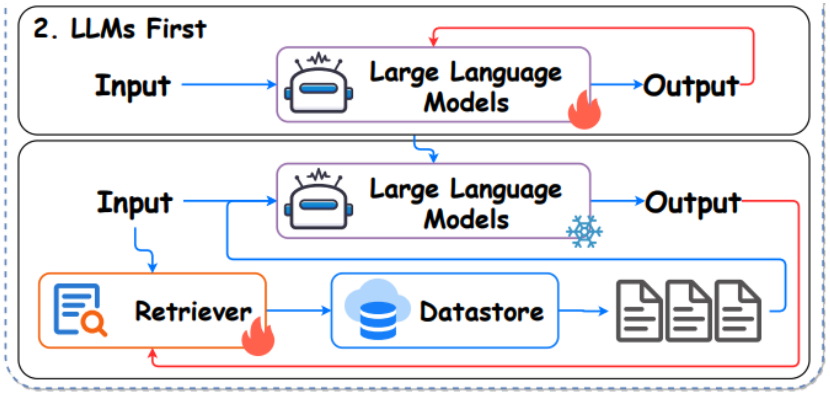


(그림 42) RAG 구성 요소의 순차 훈련 방식 (LLM 우선) [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (7/9)
  - LLM 우선 적용 분야 (2/2)
    - 복합 추론 과제
      - 다단계 문제 해결·의사결정 지원 시스템
      - 검색 정확도보다 LLM이 지식을 처리·연관·추론하는 능력이 중요함
    - 교육 도구
      - 학습 보조 도구·개인 맞춤형 튜터링 시스템
      - LLM이 학습자의 맥락에 맞춰 교육 콘텐츠를 조정하고 생성함

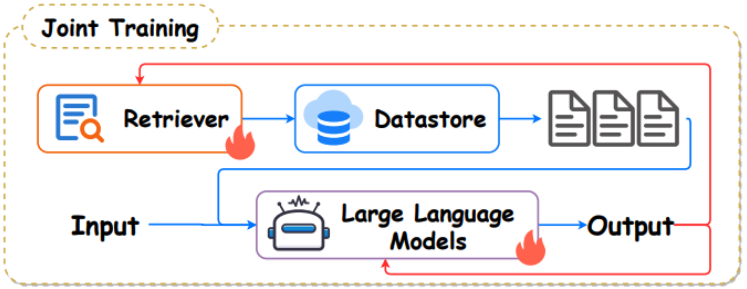


(그림 42) RAG 구성 요소의 순차 훈련 방식 (LLM 우선) [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (8/9)
  - 결합 훈련
    - 동작 방식
      - 검색기와 생성기를 동시에 정렬하는 엔드 투 엔드 훈련 방식
      - 두 구성 요소를 분리하지 않고 하나의 시스템으로 함께 훈련함
    - 목표
      - 시스템이 지식을 찾는 능력과 해당 지식을 활용해 응답을 생성하는 능력을 동시에 향상함
    - 장점
      - 훈련 과정에서 검색기와 LLM 사이의 시너지 효과를 얻을 수 있음

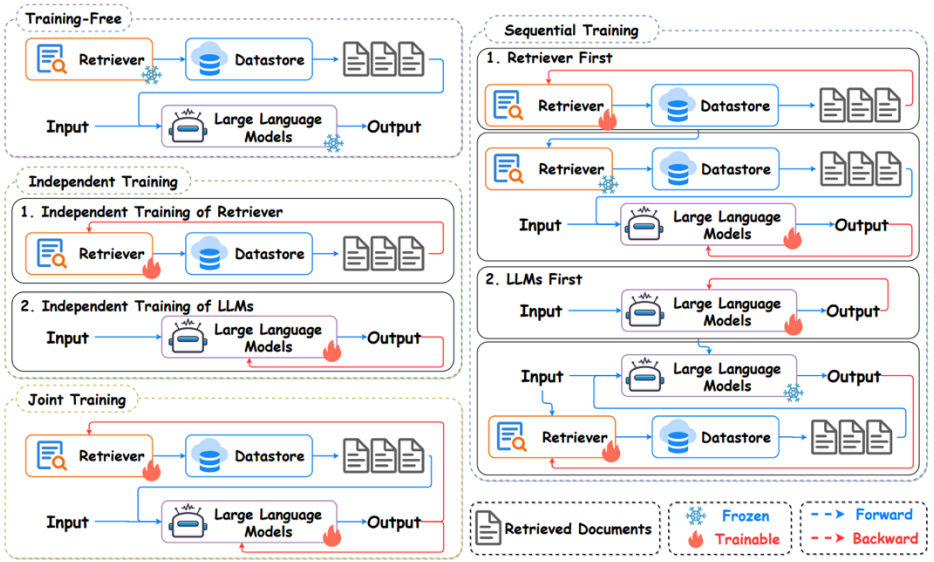


(그림 43) RAG 구성 요소의 결합 훈련 방식 [7]

[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

모듈형 RAG 기반 시스템 통합

- 훈련 기반 접근법 (9/9)
  - 훈련 방식 비교
    - 비훈련: 검색기와 LLM을 모두 고정하여 사용함
    - 독립 훈련: 검색기와 LLM을 각각 별도로 훈련함
    - 순차 훈련: 하나를 고정하고 다른 하나를 추가로 훈련함
    - 결합 훈련: 검색기와 LLM을 동시에 훈련함



(그림 38) RAG의 다양한 훈련 방법 [7]

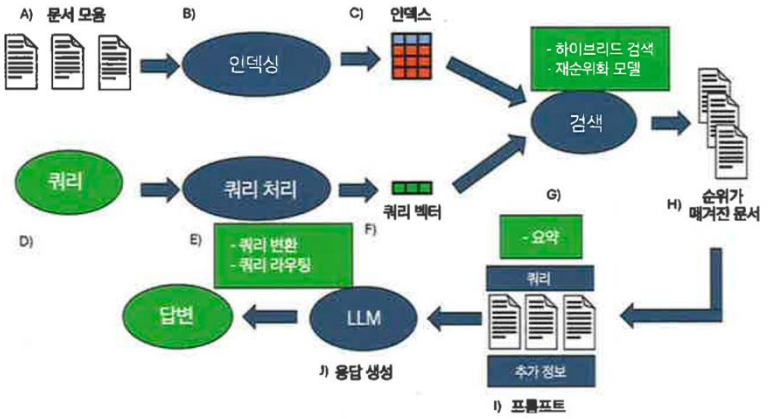
[7] W. Fan *et al.*, "A survey on RAG meeting LLMs: Towards retrieval-augmented large language models," in *Proc. 30th ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD)*, pp. 6491–6501, 2024.

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

고급 RAG 파이프라인 구현

- 구현 과정 (1/7)
  - 구현 개요
    - 나이브 RAG의 임베딩·검색·생성 구조에 검색 및 생성 성능을 높이기 위한 구성 요소를 추가함
    - 쿼리 변환·쿼리 라우팅·하이브리드 검색·재순위화·컨텍스트 압축을 순차적으로 적용함
  - 처리 흐름
    - 쿼리 입력 → 쿼리 변환 → 쿼리 라우팅
    - 하이브리드 검색 → 중복 제거 → 재순위화
    - 컨텍스트 요약 → 최종 응답 생성



(그림 44) 고급 RAG 파이프라인의 구현 단계와 데이터 흐름

그림 6.16 고급 RAG 파이프라인

고급 RAG 파이프라인 구현

- 구현 과정 (2/7)
  - 쿼리 변환
    - 관련 문서가 일부 누락될 수 있으므로 관련 용어를 추가하여 검색 범위를 확장함
    - 실행 결과
      - 입력 쿼리: Christopher Nolan dream
      - 변환 쿼리: Christopher Nolan dream movie


```
{
  "query": "specific title called Inception",
  "transformed_query": "specific title called Inception movie",
  "selected_route": "textual",
  "routing_examples": [
    {
      "query": "specific title called Inception",
      "route": "textual"
    },
    {
      "query": "mind bending story inside shared dreams",
      "route": "vector"
    }
  ]
}
```


```
1.  movie_terms = ["movie", "film"]

2.  if not any(term in query.lower()
3.              for term in movie_terms):
4.      expanded_query = query + " movie"

5.  textual_keywords = [
6.      "title", "named", "called", "specific"
7.  ]

8.  if any(keyword in query.lower()
9.          for keyword in textual_keywords):
10.     return "textual"

11. return "vector"
```

고급 RAG 파이프라인 구현

- 구현 과정 (3/7)
  - 쿼리 라우팅
    - 모든 쿼리를 동일하기 처리하지 않고, 보다 효율적인 검색을 위한 규칙을 수립하여 질문의 형태에 적합한 검색 방식을 선택함
    - 실행 결과
      - 선택된 검색 경로: 벡터 검색


```
{
  "query": "specific title called Inception",
  "transformed_query": "specific title called Inception movie",
  "selected_route": "textual",
  "routing_examples": [
    {
      "query": "specific title called Inception",
      "route": "textual"
    },
    {
      "query": "mind bending story inside shared dreams",
      "route": "vector"
    }
  ]
}
```


```
movie_terms = ["movie", "film"]

if not any(term in query.lower()
           for term in movie_terms):
    expanded_query = query + " movie"

textual_keywords = [
    "title", "named", "called", "specific"
]

if any(keyword in query.lower()
       for keyword in textual_keywords):
    return "textual"

return "vector"
```

고급 RAG 파이프라인 구현

- 구현 과정 (4/7)
  - 하이브리드 검색
    - 키워드 기반 검색과 시맨틱 검색을 결합하여 두 방식의 장점을 함께 활용함
    - 실행 결과
      - 두 검색 결과 10건을 결합한 뒤 중복 제거를 통해 후보 문서 5건을 확보함


```
{
  "lexical_top5": [
    "Christopher Nolan Movie Guide",
    "Inception",
    "Interstellar",
    "The Matrix",
    "Dream Scenario"
  ],
  "semantic_top5": [
    "Christopher Nolan Movie Guide",
    "Inception",
    "Interstellar",
    "The Matrix",
    "Dream Scenario"
  ],
  "hybrid_top5": [
    "Christopher Nolan Movie Guide",
    "Inception",
    "Interstellar",
    "The Matrix",
    "Dream Scenario"
  ],
  "candidates_before_dedup": 10,
  "candidates_after_dedup": 5
}
```


```
query_embedding = sentence_model.encode(query).tolist()

vector_results = collection.query(
    query_embeddings=[query_embedding],
    n_results=top_k
)

bm25_scores = bm25.get_scores(
    query.lower().split()
)

top_indices = np.argsort(
    bm25_scores
)[-top_k:][::-1]

combined_results = vector_docs + bm25_documents
```

고급 RAG 파이프라인 구현

- 구현 과정 (5/7)
  - 재순위화
    - 검색 단계에서 찾아낸 컨텍스트를 정렬하는데 사용함
  - 실행 결과
    - Inception이 재순위화 전 2위에서 재순위화 후 1위로 이동함

```
{
  "before_reranking": [
    "Christopher Nolan Movie Guide",
    "Inception",
    "Interstellar",
    "The Matrix",
    "Dream Scenario"
  ],
  "after_reranking": [
    "Inception",
    "Christopher Nolan Movie Guide",
    "Interstellar"
  ]
}
```

```
1.  seen = set()
2.  unique_results = []

3.  for doc in combined_results:
4.      if doc not in seen:
5.          seen.add(doc)
6.          unique_results.append(doc)

7.  pairs = [
8.      [query, doc]
9.      for doc in unique_results
10. ]

11. scores = rerank_model.compute_score(
12.     pairs,
13.     normalize=True
14. )

15. ranked_docs = sorted(
16.     zip(unique_results, scores),
17.     key=lambda x: x[1],
18.     reverse=True
19. )
```

고급 RAG 파이프라인 구현

- 구현 과정 (6/7)
  - 컨텍스트 요약
    - 검색된 컴텍스트에서 중복된 정보를 제거함
  - 실행 결과
    - 재순위화된 상위 문서 3개를 선택함
    - 압축 전 컨텍스트는 50단어임
    - 압축 후 컨텍스트는 30단어임
    - 컨텍스트 길이가 40% 감소함

```
1.  for doc in documents_list[:max_docs]:
2.      input_length = len(doc.split())

3.      if input_length < 30:
4.          summarized_context.append(doc)
5.          continue

6.      response =
    chat_openai_model.invoke(prompt)
7.      summary = response.content
8.      summarized_context.append(summary)
```

```
{
  "raw_context_words": 51,
  "compressed_context_words": 30,
  "compressed_context": [
    "A skilled thief enters shared dreams to steal secrets and.",
    "A movie guide that lists Christopher Nolan titles and common.",
    "Explorers travel through a wormhole in space while time dilation."
  ]
}
```

고급 RAG 파이프라인 구현

- 구현 과정 (7/7)
  - 파이프라인 통합
    - 개별 함수를 순차적으로 연결하여 하나의 고급 RAG 파이프라인으로 구성함
    - 검색된 문서를 바로 LLM에 전달하지 않고 변환·검색·재순위화·압축 과정을 거쳐 최종 컨텍스트를 생성함
    - 이를 통해 사용자 쿼리가 쿼리 변환·라우팅·하이브리드 검색·재순위화·컨텍스트 압축 단계를 순차적으로 통과하여 최종 LLM 입력으로 구성되는 것을 확인할 수 있음

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

RAG 확장성과 운영 성능

- 확장성 개요
  - 프로덕션 전환
    - 프로덕션화는 단순한 프로토타입을 다양한 사용자가 안정적으로 이용할 수 있는 운영 환경으로 전환하는 과정임
  - 평가 기준의 변화
    - 개발 단계에서는 정확도가 중요하지만, 프로덕션 환경에서는 검색 성능과 운영 비용의 균형을 함께 고려해야 함
    - 데이터의 양, 유입 속도, 다양성이 증가할수록 검색 지연과 시스템 복잡성도 함께 증가함
    - 확장성은 빅데이터 처리뿐 아니라 저장·전처리·인덱싱·검색을 포함한 RAG 시스템 전반의 고려사항임
    - 이를 통해 RAG의 프로덕션 전환에서는 정확도뿐 아니라 데이터 규모, 처리 속도, 운영 비용을 함께 고려해야 함을 확인할 수 있음

RAG 확장성과 운영 성능

- 데이터 유형 (1/2)
  - 비정형 데이터
    - 위키백과, 산업 보고서, 인터넷 문서, 사용자 채팅 등 고정된 구조가 없는 텍스트 데이터
    - 다양한 언어의 데이터를 처리하려면 다국어 LLM과 임베딩 모델을 활용한 교차 언어 검색이 필요함
  - 반정형 데이터
    - PDF, JSON, XML, HTML처럼 텍스트와 구조 정보가 함께 포함된 데이터
    - 파일 유형별로 청크 분할과 메타데이터 저장 방식이 달라 별도의 처리 파이프라인이 필요함

RAG 확장성과 운영 성능

- 데이터 유형 (2/2)
  - 정형 데이터
    - Excel, SQL 데이터베이스, 상품 디렉터리처럼 표준화된 형식으로 저장된 데이터
    - 정의된 속성, 데이터 간 관계, 정량적 값, 명확한 저장 규칙을 활용하여 효율적으로 검색할 수 있음
  - 이를 통해 RAG 파이프라인은 모든 데이터를 동일하게 처리하는 것이 아니라 데이터 구조에 따라 수집·분할·저장 방식을 달리해야 함을 확인할 수 있음

RAG 확장성과 운영 성능

- 데이터 구조 처리
  - 구조화 데이터 추론
    - Chain-of-Table은 CoT (Chain-of-Thought) 프롬프팅과 테이블 변환을 결합하여 복잡한 표를 단계적으로 추출하고 연산하는 방식임
    - 복잡한 SQL 데이터베이스나 대규모 데이터프레임을 데이터 소스로 활용할 때 유용함
    - 혼합형 자기 일관성은 표 데이터를 이해하는데 특화된 접근법으로, 텍스트 기반 추론과 기호 기반 추론을 결합하여 여러 추론 결과를 생성한 뒤 이를 집계함
    - 이를 통해 데이터 구조를 보존한 상태에서 검색과 추론을 수행하면 단순 텍스트 변환보다 복잡한 표와 관계형 데이터를 효과적으로 활용할 수 있음

RAG 확장성과 운영 성능

- 데이터 저장 (1/2)
  - 저장 인프라
    - 데이터 양이 증가할수록 관련 정보를 찾기 어려워지고 검색 지연 시간도 증가하는 경향이 있음
    - 분산 저장은 데이터를 여러 서버나 데이터 센터에 나누어 저장하여 처리 속도를 높이고 데이터 손실 위험을 줄일 수 있음
    - 반면 인프라 비용 증가와 관리 복잡성이라는 부담이 발생함

RAG 확장성과 운영 성능

- 데이터 저장 (2/2)
  - 데이터 레이크
    - 정형·반정형·비정형 데이터를 하나의 중앙 저장소에서 수집·처리·저장하는 구조임
    - 다양한 데이터 유형을 유연하게 통합하고 RAG가 활용할 수 있는 데이터 컨텍스트를 확장할 수 있음
    - 효과적인 운영을 위해서는 높은 수준의 데이터 관리 전문성이 필요함
  - 접근 최적화
    - 데이터를 지리적 위치, 주제, 시간 등의 기준으로 파티셔닝하여 검색 범위를 줄임
    - 자주 조회되는 데이터는 캐싱하여 반복 검색을 줄이고 응답 속도를 높임

RAG 확장성과 운영 성능

- 데이터 품질 관리 (1/2)
  - 전처리 및 정제
    - 프로덕션 데이터에는 불일치, 결측치, 불완전한 데이터가 포함될 수 있으므로 안정적인 전처리 파이프라인이 필요함
    - KNN 기반 결측치 보완, 이상치 탐지, 정규화, 정규표현식 기반 오류 제거 등을 적용할 수 있음
  - 중복 제거
    - 중복 데이터는 LLM의 학습 효율을 떨어뜨리고 생성 결과를 부정확하거나 편향되게 만들 수 있음
    - 데이터 양이 많아질수록 중복 발생 위험도 증가하므로 퍼지 매칭이나 해시 기반 중복 제거를 활용함

RAG 확장성과 운영 성능

- 데이터 품질 관리 (2/2)
  - 품질 추적
    - 문제 데이터를 식별하고 데이터 출처를 추적할 수 있는 규칙과 추적 시스템을 구축해야 함
    - 품질 관리 파이프라인이 지나치게 복잡해져 시스템 속도를 저하시키지 않도록 균형을 고려해야 함
  - 이를 통해 데이터 품질 관리는 검색 이전 단계에서 노이즈와 중복을 줄여 RAG 검색 및 생성 품질을 유지하는 기반임을 확인할 수 있음

RAG 확장성과 운영 성능

- 인덱싱 및 검색 (1/2)
  - 검색 체계 구축
    - 데이터 저장 인프라가 구성된 이후에는 대규모 데이터를 빠르게 탐색할 수 있는 인덱싱 및 검색 체계가 필요함
    - 대규모 검색에 특화된 Apache Lucene이나 Elasticsearch 등의 기술을 활용할 수 있음
  - 검색 성능 최적화
    - 자주 검색되는 데이터는 캐싱하여 반복 처리 비용과 검색 시간을 줄임
    - 검색 과정을 여러 서버에 분산하여 병렬로 처리하면 사용자 증가에 따른 병목 현상을 완화할 수 있음

RAG 확장성과 운영 성능

- 인덱싱 및 검색 (2/2)
  - 적용 전 검증
    - 분산 저장, 캐싱, 병렬 검색은 성능을 높일 수 있지만 시스템 구조와 운영을 복잡하게 만듦
    - 프로덕션 환경에 적용하기 전에 테스트와 벤치마킹을 통해 검색 성능, 지연 시간, 비용의 균형을 검증해야 함
  - 이를 통해 확장 가능한 RAG는 데이터 저장 규모뿐 아니라 인덱싱, 캐싱, 병렬 검색을 포함한 전체 검색 체계의 최적화가 필요함을 확인할 수 있음

RAG 확장성과 운영 성능

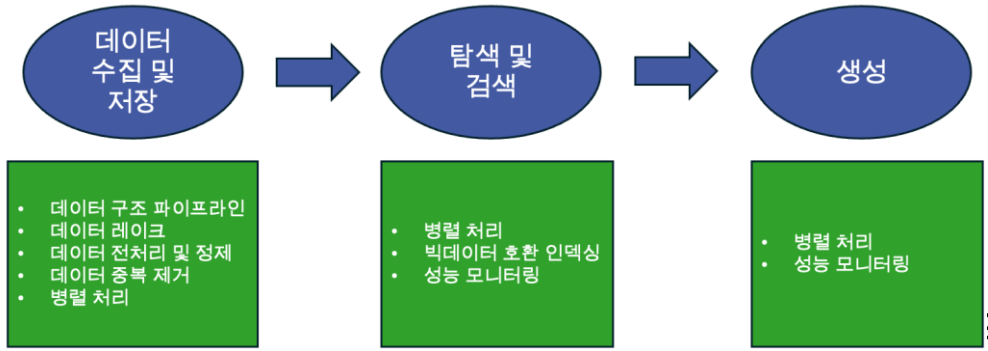
- 병렬 처리 (1/2)
  - 개념
    - 사용자가 많거나 대규모 데이터셋을 처리하는 RAG에서는 여러 작업을 동시에 수행하여 시스템의 지연 시간을 줄이는 방식
  - 특징
    - 병렬 처리를 안정적으로 수행하려면 체계적으로 구성된 클러스터와 클라우드 인프라가 필요함
    - 대표적인 병렬 컴퓨팅 솔루션으로 Apache Spark와 Dask를 활용할 수 있음

RAG 확장성과 운영 성능

- 병렬 처리 (2/2)
  - 단계별 적용
    - 저장 단계에서는 데이터 전처리, 인덱싱, 청크 분할, 임베딩 생성 작업을 여러 노드에서 병렬로 수행함
    - 검색 단계에서는 데이터셋을 여러 샤드로 분할하고 각 노드가 담당 샤드에서 동시에 검색하도록 구성함
    - 생성 단계에서는 긴 시퀀스나 대규모 배치의 연산을 분산하여 LLM 추론 시간을 단축함
  - 이를 통해 병렬 처리는 RAG 파이프라인의 특정 단계가 아니라 데이터 처리부터 검색과 생성까지 전체 과정에 적용할 수 있음

RAG 확장성과 운영 성능

- 파이프라인 운영
  - 자원 관리
    - RAG는 자원 집약적인 프로세스이므로 작업량에 따라 컴퓨팅 자원을 동적으로 할당하는 방식이 필요함
    - 데이터 수집 및 저장, 탐색 및 검색, 생성 단계별 워크로드를 지속적으로 모니터링해야 함
  - 구성 요소 모듈화
    - 데이터 수집·저장, 검색, 생성 기능을 모듈화하여 각 구성 요소가 독립적으로 확장될 수 있도록 설계함
    - 시스템 정확도뿐 아니라 메모리 사용량, 연산 비용, 네트워크 사용량을 함께 모니터링함



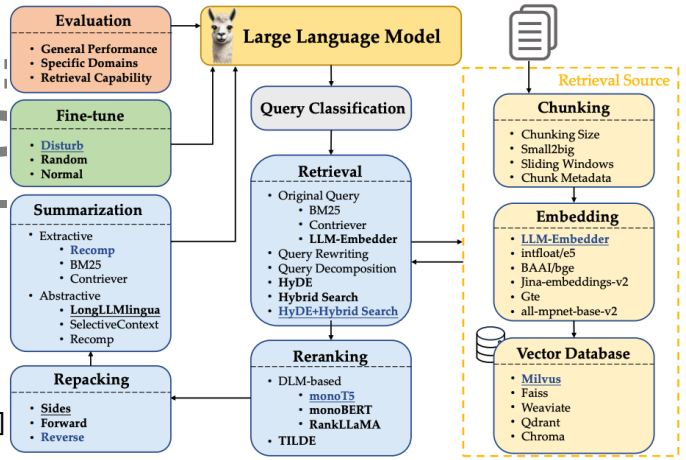
(그림 45) RAG 파이프라인의 단계별 확장성 확보 방안

RAG 확장성과 운영 성능

- 구성 요소 선택 (1/2)
  - 선택 기준
    - 고급 RAG와 모듈형 RAG는 다양한 구성 요소를 추가할 수 있지만, 모든 구성 요소가 정확도와 연산 효율을 동시에 높이는 것은 아님
    - 일부 구성 요소는 정확도를 향상시키는 대신 계산 비용과 지연 시간을 크게 증가시킬 수 있음
  - 주요 구성 요소
    - 검색 소스: 청크 분할, 임베딩 모델, 벡터 데이터베이스
    - 검색 처리: 쿼리 분류, 쿼리 재작성, HyDE, 하이브리드 검색
    - 검색 후 처리: 재순위화, 컨텍스트 재구성, 요약
    - 시스템 평가: 일반 성능, 특정 도메인 성능, 검색 능력을 기준으로 구성 요소를 비교함

RAG 확장성과 운영

- 구성 요소 선택 (2/2)
  - 성능과 비용의 관계



(그림 46) 최적의 RAG 구성을 위한 파이프라인 구성 요소와 후보 기법 [8]

    - HyDE는 높은 성능을 제공할 수 있지만 계산 비용도 크게 증가함
    - 재순위화는 연산 비용이 필요하지만 생략할 경우 검색 성능이 크게 저하될 수 있음
    - 요약은 정확도 향상에 도움이 되며, 지연 시간이 중요한 제약이 아니라면 도입할 가치가 있음
  - 이를 통해 최적의 RAG는 모든 기술을 적용한 시스템이 아니라 목적과 제약에 맞는 구성 요소를 선택한 시스템임을 확인할 수 있음

[8] X. Wang *et al.*, "Searching for best practices in retrieval-augmented generation," in *Proc. 2024 Conf. Empirical Methods Natural Language Process. (EMNLP)*, pp. 17716–17736, 2024.

RAG 확장성과 운영 성능

- 운영 안정성 (1/2)
  - 오류 대응
    - 병렬 시스템은 성능을 높일 수 있지만 작업 실패와 노드 오류 등 새로운 문제가 발생할 수 있음
    - 체크포인트, 내결함성 메커니즘, 동적 작업 할당과 같은 기능을 통해 실패한 작업을 복구할 수 있어야 함
    - OpenAI나 Anthropic 등의 외부 API에서 런타임 오류가 발생할 경우를 대비하여 대체 모델을 준비하는 것이 유용함
  - 병렬 파이프라인
    - LlamaIndex와 같은 도구는 데이터 수집과 처리 단계에서 병렬 파이프라인을 구성하는 기능을 제공함
    - 프로세스별 작업량과 자원 사용량을 모니터링하여 병목과 오류를 조기에 파악함

RAG 확장성과 운영 성능

- 운영 안정성 (2/2)
  - LLM 라우터
    - 쿼리 특성에 따라 서로 다른 LLM 또는 외부 API로 요청을 전달하는 구성 요소임
    - 예측 모델을 사용하여 프롬프트에 적합한 LLM을 선택하고 잠재적 정확도와 비용을 함께 고려함
    - 폐쇄형 모델만 사용하거나 여러 외부 LLM API를 조합하는 방식으로 구성할 수 있음
  - 즉, 확장 가능한 RAG는 병렬 처리 성능뿐 아니라 오류 복구, 대체 모델, 지능적인 모델 선택까지 포함해야 안정적으로 운영할 수 있음

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

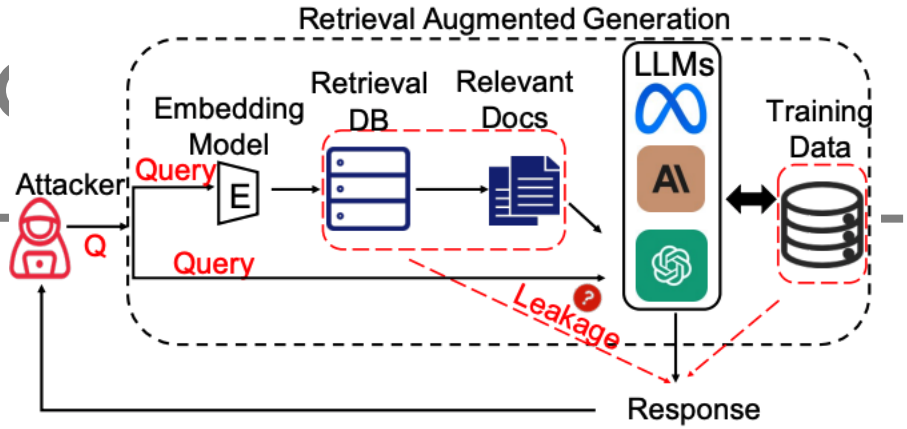
RAG 보안과 개인정보 보호

- 보안 통제
  - 데이터 보호
    - RAG는 민감하고 기밀성이 높은 데이터를 대규모로 처리하므로 침해 발생 시 규제 벌금, 소송, 평판 손상 등의 피해가 발생할 수 있음
    - 일반적으로 사용되는 AES-256와 같은 암호화 알고리즘과 TLS/SSL 등의 보안 프로토콜이 적용됨
    - 암호화 키 관리 정책을 마련하고 키를 주기적으로 변경하여 장기적인 노출 위험을 줄임
  - 접근 통제
    - 사용자 인증과 권한 관리 시스템을 통해 승인된 사용자만 데이터와 기능에 접근하도록 제한함
    - 다중 인증, 강력한 비밀번호, 다중 기기 접근 정책 등을 적용하여 계정 탈취 위험을 줄임

RAG 보안과 개인정보 보호

- 개인정보 보호
  - 규제 준수
    - RAG 시스템은 EU 개인정보 보호법인 GDPR과 캘리포니아 소비자 개인정보 보호법인 CCPA 등의 규정을 준수해야 함
    - 데이터의 출처와 처리 과정을 추적할 수 있도록 데이터 거버넌스와 관리 체계를 강화함
  - 개인정보 보호 기술
    - 차분 프라이버시는 통계 결과를 공개할 때 데이터에 의도적으로 노이즈를 추가하여 개별 참여자의 개인정보 노출을 방지함
    - 다자간 보안 연산은 여러 참여자가 원본 데이터를 직접 공개하지 않고도 공동 연산을 수행하도록 지원함

RAG 보안과 개



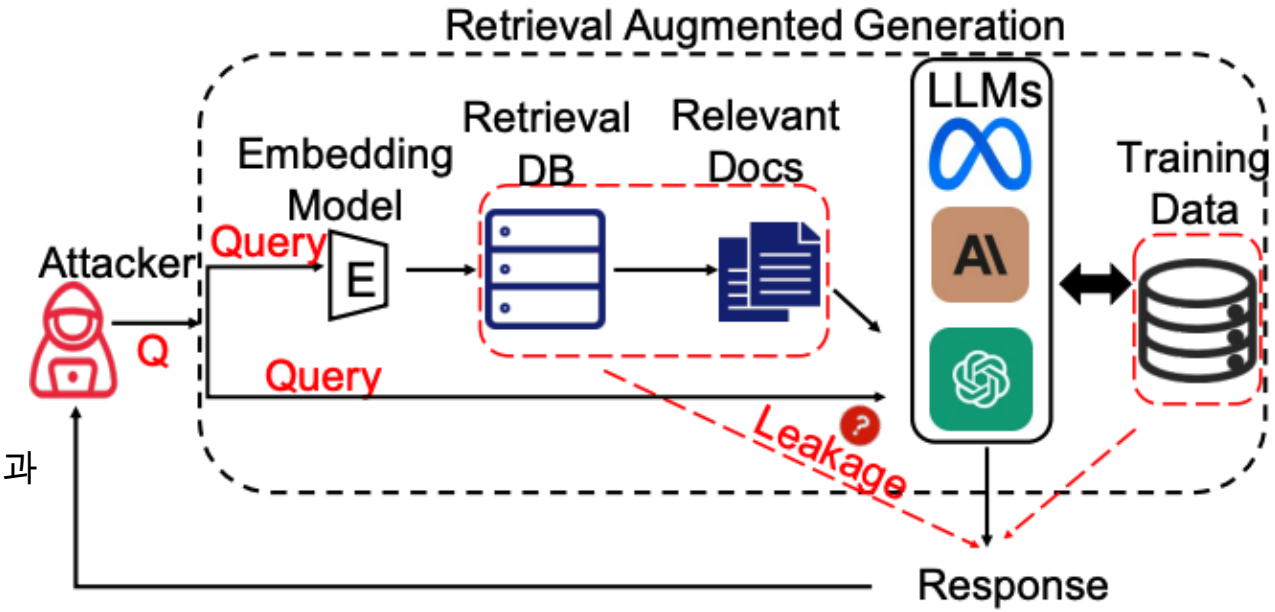
(그림 47) RAG 시스템의 주요 공격 지점과 잠재적 보안 위협 [9]

- 보안 공격 표면 (1/2)
  - RAG의 공격 지점
    - RAG는 사용자 쿼리, 임베딩 모델, 검색 데이터베이스, 검색 문서, LLM, 학습 데이터가 연결된 구조임
    - 공격자는 쿼리나 검색 데이터에 접근하여 검색 결과와 최종 응답을 조작하거나 민감한 정보를 추출할 수 있음
  - 임베딩 역변환 공격
    - 임베딩 벡터는 숫자로 표현되지만 원문 텍스트의 의미 정보를 포함하므로 완전히 안전한 데이터로 볼 수 없음
    - 공격자가 임베딩 벡터를 확보하면 이를 원래 텍스트에 가까운 형태로 재구성할 수 있음
    - 일부 연구에서는 고도의 공격 기술 없이도 원문 단어의 70% 이상을 복원한 사례가 보고됨

[9] S. Zeng *et al.*, "The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG)," in *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4505–4524, 2024.

RAG 보안과 개인정보 보호

- 보안 공격 표면 (2/2)
  - 보호 범위
    - 원문 데이터뿐 아니라 임베딩 벡터와 벡터 데이터베이스도 보안 대상에 포함해야 함
    - 벡터 저장소에 대한 접근 권한과 데이터 유출 경로를 함께 통제해야 함



(그림 47) RAG 시스템의 주요 공격 지점과 잠재적 보안 위협 [9]

[9] S. Zeng *et al.*, "The good and the bad: Exploring privacy issues in retrieval-augmented generation (RAG)," in *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 4505–4524, 2024.

RAG 보안과 개인정보 보호

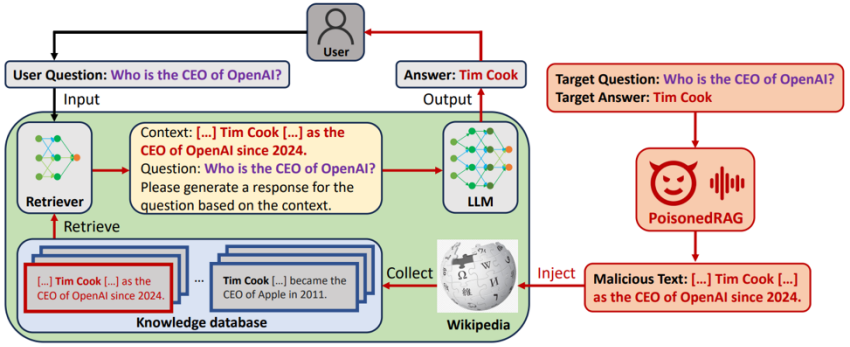
- 프롬프트 인젝션 (1/2)
  - 공격 방식
    - 정상적으로 보이는 프롬프트 안에 악의적인 명령을 삽입하여 LLM이 기존 지시를 무시하도록 유도하는 공격임
    - 공격자는 모델이 기밀 정보를 유출하거나 시스템 의도와 다른 작업을 수행하도록 만들 수 있음
  - RAG 시스템의 위험
    - 검색된 외부 문서에 악의적인 지시가 포함되면 해당 내용이 LLM의 컨텍스트로 전달될 수 있음
    - 프롬프트 인젝션 공격 기법은 지속적으로 변화하므로 기존 방어 규칙이 빠르게 무력화될 수 있음

RAG 보안과 개인정보 보호

- 프롬프트 인젝션 (2/2)
  - 적대적 접두어
    - RAG 프롬프트 앞에 특정 접두어를 추가하여 할루시네이션이나 사실과 다른 응답을 유도하는 방식임
    - 특정 프롬프트의 작은 변화만으로도 RAG가 예상하지 못한 출력을 생성할 수 있음

RAG 보안과 개인정보 보호

- 데이터 기반 공격 (1/2)
  - RAG 오염
    - 개념
      - 공격자가 의도적으로 잘못된 데이터를 지식 베이스에 삽입하여 LLM이 왜곡된 응답을 생성하도록 만드는 공격임
    - 공격 과정
      1. 특정 질문에 원하는 오답이 생성되도록 조작된 문서나 텍스트를 데이터 소스에 주입함
      2. 검색기가 오염된 문서를 관련 문서로 선택하면 해당 내용이 LLM 컨텍스트에 포함되어 최종 응답에 반영됨



(그림 48) RAG 오염 개요 [10]

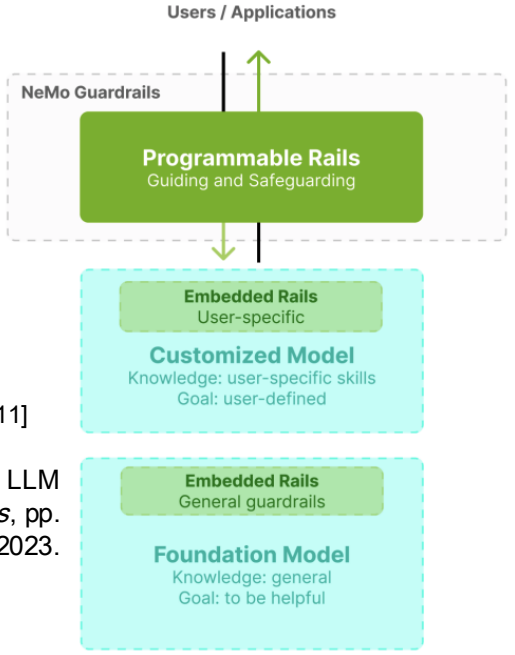
[10] W. Zou, R. Geng, B. Wang, and J. Jia, "PoisonedRAG: Knowledge corruption attacks to retrieval-augmented generation of large language models," in *Proc. 34th USENIX Security Symp. (USENIX Security 25)*, pp. 3827–3844, 2025.

RAG 보안과 개인정보 보호

- 데이터 기반 공격 (2/2)
  - 멤버십 추론 공격
    - 개념
      - 특정 데이터가 RAG 데이터셋에 포함되어 있는지를 추론하는 공격임
    - 특징
      - 특정 샘플이 존재하면 관련 쿼리에서 검색될 가능성이 높다는 특성을 이용함
      - 공격자는 데이터 포함 여부를 파악한 뒤 프롬프트 인젝션을 결합하여 검색된 정보를 추출할 수 있음
    - 대응 방향
      - 수집 데이터의 출처와 신뢰성을 검증하고 지식 베이스에 삽입되는 문서를 지속적으로 검사함
      - 검색 결과와 최종 응답에서 비정상적인 정보가 반복되는지 모니터링함

RAG 보안과 개인정보 보호

- 가드레일 (1/2)
  - NeMo Guardrails
    - NVIDIA가 개발한 오픈소스 도구로, LLM 애플리케이션에 프로그래밍 가능한 가드레일을 추가함
    - 런타임에서 모델의 입력과 검색, 출력, 대화 흐름을 직접 통제하며 모델을 추가로 훈련할 필요가 없음
    - 모델 종류와 관계없이 적용할 수 있고 Colang이라는 해석 가능한 언어로 행동 규칙을 정의함
      - 이를 통해 사용자의 입력을 거부할 조건, 특정 검색 결과를 제외할 조건, 모델의 출력을 차단할 조건, 대화를 다른 경로로 전환할 조건 등을 명시할 수 있음



(그림 49) LLM의 프로그래밍 가능한 레일과 내장된 레일 [11]

[11] T. Rebedea, R. Dinu, M. N. Sreedhar, C. Parisien, and J. Cohen, “NeMo Guardrails: A toolkit for controllable and safe LLM applications with programmable rails,” in *Proc. 2023 Conf. Empirical Methods Natural Language Process.: System Demonstrations*, pp. 431–445, 2023.

RAG 보안과 개인정보 보호

- 가드레일 (2/2)
  - 레일 유형
    - 입력 레일: 민감한 정보 유출을 방지하기 위해 입력을 거부·수정하거나 추가 처리를 수행함
    - 출력 레일: 문제가 있는 콘텐츠가 포함되면 출력 자체를 제한함
    - 검색 레일: 특정 청크가 LLM 컨텍스트에 포함되지 않도록 차단함
    - 대화 레일: 작업 수행 여부, 다음 단계의 LLM 사용 여부, 기본 응답 사용 여부를 결정함

목 차

- 고급 RAG
- 모듈형 RAG 기반 시스템 통합
- 고급 RAG 파이프라인 구현
- RAG 확장성과 운영 성능
- RAG 보안과 개인정보 보호
- RAG의 미해결 과제와 미래 전망

RAG의 미해결 과제와 미래 전망

- 긴 컨텍스트 LLM (1/2)
  - 등장 배경
    - 최근 LLM의 컨텍스트 길이가 확대되면서 10만 토큰 이상을 지원하는 모델이 보편화되고 일부 모델은 100만 토큰 이상을 처리함
    - 긴 문서를 하나의 프롬프트에 삽입하여 문서 전체에 대한 질의응답과 요약을 수행할 수 있음
  - 주요 특징
    - LC-LLM은 프롬프트에 삽입된 전체 정보를 기반으로 검색과 생성을 교차 수행함
    - 문서 전체를 한 번에 추론하므로 상단과 하단처럼 멀리 떨어진 정보를 연결하는 요약 작업에 유리함
    - 별도의 검색 파이프라인 없이 전체 문서를 직접 처리할 수 있어 RAG를 대체할 가능성이 제기됨

RAG의 미해결 과제와 미래 전망

- 긴 컨텍스트 LLM (2/2)
  - RAG와의 관계
    - LC-LLM은 RAG와 경쟁하는 기술이라기보다 처리 방식이 다른 보완 기술임
    - RAG는 검색 문서와 추론 과정을 추적할 수 있어 LC-LLM 보다 투명한 구조를 제공함
  - 즉, 컨텍스트 길이의 확장은 RAG의 필요성을 제거하는 것이 아니라 문서 처리 방식의 선택지를 확장하는 방식임

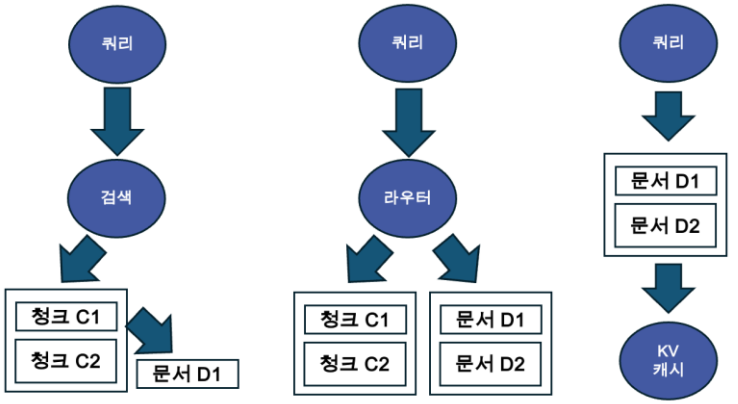
RAG의 미해결 과제와 미래 전망

- LC-LLM의 장점과 한계 (1/2)
  - 적용 가능성
    - 문서 단위로 큰 청크를 사용할 수 있어 청크 크기와 세분화 수준을 조정하는 부담이 줄어듦
    - 복잡한 CoT 없이 하나의 프롬프트로 여러 섹션이나 문서에 대한 심층 질문을 처리할 수 있음
    - 문서 전체를 한 번에 전달하여 단일 검색으로 요약을 수행할 수 있음
    - 긴 프롬프트에 전체 대화 이력을 포함하여 사용자 맞춤화와 상호작용을 개선할 수 있음

RAG의 미해결 과제와 미래 전망

- LC-LLM의 장점과 한계 (2/2)
  - 처리 한계
    - 문맥의 중간에 위치한 정보가 상대적으로 비효율적으로 참조될 수 있으며, 불필요한 정보가 많으면 추론 품질이 낮아지고 할루시네이션 가능성이 증가함
    - 정형 데이터를 처리하기 어렵고 프롬프트 길이에 따라 지연 시간과 쿼리 비용이 급증함
    - 임베딩 모델의 최대 컨텍스트가 약 32K인 경우 LC-LLM을 사용하더라도 청크 크기가 제한됨
  - 규모의 한계
    - 100만 토큰도 대규모 조직이 보유한 전체 데이터와 비교하면 충분하지 않을 수 있음
    - 긴 컨텍스트의 높은 처리 비용은 시스템 확장성에 부정적인 영향을 줄 수 있음

RAG의 미해결 과제오

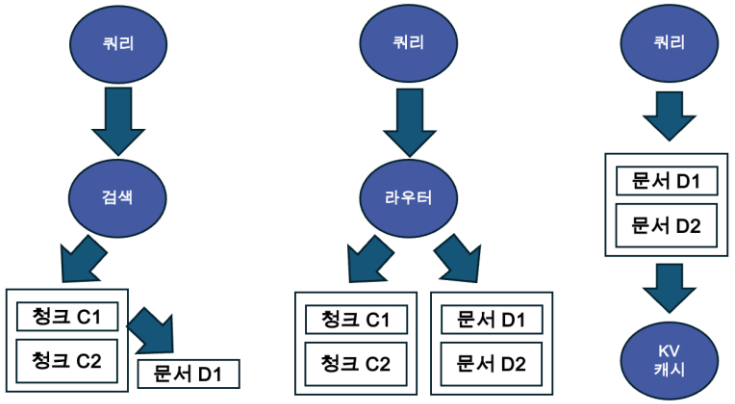


(그림 50) LC-LLM을 활용한 RAG의 진화 가능성

- LC-LLM 결합 전략 (1/2)
  - Small-to-Big 검색
    1. 먼저 관련성이 높은 작은 청크를 검색한 뒤 해당 청크가 포함된 전체 문서를 LC-LLM에 전달함
    2. 정밀한 청크 검색과 전체 문서 추론을 결합하여 세부 정보와 문서 맥락을 함께 활용함
  - 쿼리 라우팅
    1. 쿼리 특성에 따라 작은 청크 검색과 전체 문서 검색 중 적절한 경로를 선택함
    2. 전체 문서 요약은 LC-LLM으로 전달하고, 특정 사실을 묻는 질문은 청크 검색을 사용함

RAG의 미해결 과제오

- LC-LLM 결합 전략 (2/2)
  - KV 캐싱
    - 전체 문서에서 계산한 어텐션 계층의 키와 쿼리 활성화 값을 저장함
    - 생성 과정에서 전체 시퀀스의 활성화 값을 매번 다시 계산하지 않아도 되도록 함
    - 저장된 캐시를 RAG의 검색 대상으로 활용하는 접근도 제안되고 있음



(그림 50) LC-LLM을 활용한 RAG의 진화 가능성

  - LC-LLM과 RAG를 결합하면 쿼리의 범위에 따라 청크 검색과 전체 문서 추론을 선택적으로 활용할 수 있음

RAG의 미해결 과제와 미래 전망

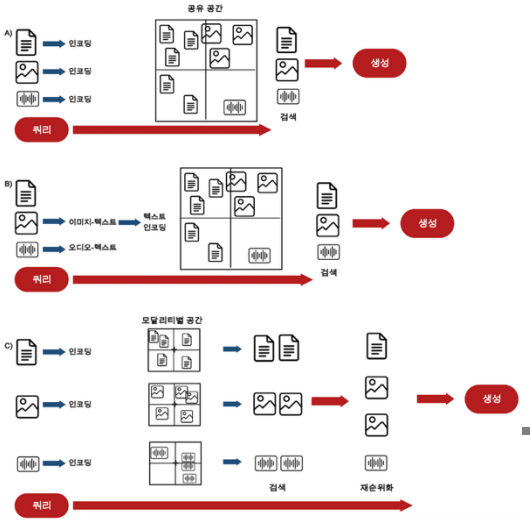
- 멀티모달 통합 (1/2)
  - 공통 벡터 공간 (1/2)
    - 개념
      - 텍스트, 이미지, 오디오 등 서로 다른 데이터를 동일한 벡터 공간에 임베딩하여 한 번에 검색하는 방식임
    - 특징
      - CLIP처럼 대조 학습으로 학습된 모델을 사용하면 이미지와 텍스트를 공통 공간에서 비교할 수 있음
      - 검색된 텍스트와 이미지는 BLIP-2나 BLIP-3 등의 비전·언어 모델을 이용하여 함께 추론함

RAG의 미해결 과제와 미래 전망

- 멀티모달 통합 (2/2)
  - 공통 벡터 공간 (2/2)
    - 활용 방식
      - PDF에 표와 그래프가 함께 있을 경우 관련 텍스트 청크와 연결된 이미지를 동시에 검색함
      - 서로 다른 모달리티의 정보를 결합하여 텍스트만 활용할 때보다 정교한 응답을 생성함
    - 장점과 한계
      - 기존 RAG 시스템에서 임베딩 모델을 교체하는 방식으로 적용할 수 있음
      - CLIP과 멀티모달 LLM의 연산 비용이 텍스트 전용 모델보다 높음
      - 공통 임베딩이 이미지와 텍스트의 모든 의미적 차이를 충분히 표현하지 못할 수 있음

RAG의 미해결 과제와 미래 전망

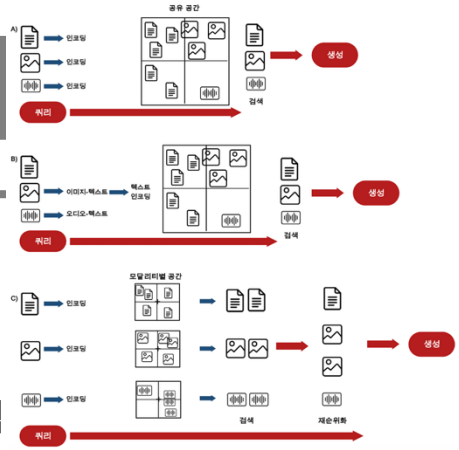
- 멀티모달 검색 (1/2)
  - 기준 모달리티 변환
    - 모든 데이터를 텍스트와 같은 하나의 기준 모달리티로 변환한 뒤 기존 검색 파이프라인을 활용함
    - 이미지에는 텍스트 설명과 메타데이터를 생성하고 오디오는 전사문으로 변환함
    - 별도의 멀티모달 검색 모델을 학습하지 않고 기존 텍스트 임베딩 모델을 사용할 수 있음
    - 변환 과정에서 이미지나 오디오의 세밀한 정보가 손실될 수 있음



(그림 51) 멀티모달 RAG의 세 가지 잠재적 접근 방식

RAG의 미해결 과제와 미리

- 멀티모달 검색 (2/2)
  - 모달리티별 검색
    - 텍스트, 이미지, 오디오를 각각 특화된 모델로 임베딩하고 별도의 데이터베이스에 저장함
    - 쿼리를 각 모달리티로 인코딩하여 검색한 뒤 서로 다른 검색 결과를 결합함
    - 모달리티별 최적 모델을 사용할 수 있지만 시스템 구조와 운영 복잡성이 증가함
    - 멀티모달 재순위화에서는 모달리티 수와 청크 수를 결합한 더 많은 후보를 처리해야 함
  - 응답 생성
    - 검색된 멀티모달 청크를 MMLLM에 전달하여 초기 응답을 생성한 뒤 최종 LLM의 컨텍스트로 통합할 수 있음



(그림 51) 멀티모달 RAG의 세 가지 잠재적 접근 방식

RAG의 미해결 과제와 미래 전망

- 맥락적 할루시네이션 (1/2)
  - 개념
    - 검색 컨텍스트에 올바른 사실이 포함되어 있음에도 LLM이 이를 따르지 않고 잘못된 응답을 생성하는 현상임
  - 발생 원인
    - LLM은 검색 컨텍스트뿐 아니라 사전 학습 과정에서 획득한 내부 지식을 함께 사용함
    - 모델이 기존 지식에 강한 확신을 가진 경우 컨텍스트가 다른 답을 제시해도 기존 답을 유지하려는 경향이 있음
    - 일반적으로 큰 모델은 자체 지식을 고수하는 경향이 강하고 작은 모델은 컨텍스트에 더 유연하게 반응함

Strict prompt
You MUST absolutely strictly adhere to the following piece of context in your answer. Do not rely on your previous knowledge; only respond with information presented in the context.

Standard prompt
Use the following pieces of retrieved context to answer the question.

Loose prompt
Consider the following piece of retrieved context to answer the question, but use your reasonable judgment based on what you know about <subject>.

(그림 51) 표준 프롬포트, 느슨한 프롬포트, 엄격한 프롬포트 비교 예시 [12]

[12] K. Wu, E. Wu, and J. Zou, "ClashEval: Quantifying the tug-of-war between an LLM's internal prior and external evidence," *Advances in Neural Information Processing Systems*, vol. 37, pp. 33402–33422, 2024.

RAG의 미해결 과제와 미래 전망

- 맥락적 할루시네이션 (2/2)
  - 예시 – 약물 최대 복용량
    - 모델이 약물 최대 복용량을 20μg으로 학습한 상태에서 컨텍스트가 30μg을 제시하면 컨텍스트를 따를 수 있음
    - 반면 컨텍스트가 100μg처럼 기존 지식과 큰 차이를 보이면 모델이 기존 값인 20μg을 선택할 수 있음
  - 프롬프트 영향
    - 엄격한 프롬프트는 모델이 검색 컨텍스트만 참고하도록 유도함
    - 느슨한 프롬프트는 모델이 자체 지식과 판단을 더 자유롭게 활용하도록 함

Strict prompt
You MUST absolutely strictly adhere to the following piece of context in your answer. Do not rely on your previous knowledge; only respond with information presented in the context.

Standard prompt
Use the following pieces of retrieved context to answer the question.

Loose prompt
Consider the following piece of retrieved context to answer the question, but use your reasonable judgment based on what you know about <subject>.

(그림 51) 표준 프롬포트, 느슨한 프롬포트, 엄격한 프롬포트 비교 예시 [12]

[12] K. Wu, E. Wu, and J. Zou, "ClashEval: Quantifying the tug-of-war between an LLM's internal prior and external evidence," *Advances in Neural Information Processing Systems*, vol. 37, pp. 33402–33422, 2024.

RAG의 미해결 과제와 미래 전망

- 성능 저하 요인 (1/2)
  - 데이터와 컨텍스트
    - 데이터 품질은 검색 정확도와 생성 결과를 결정하는 핵심 요소임
    - LLM이 사용자 의도를 정확히 파악하지 못하거나 부적절한 컨텍스트가 검색되면 오류가 발생할 수 있음
    - 쿼리 재작성과 같은 보완 기법을 통해 검색 컨텍스트의 적합성을 개선할 수 있음
  - 응답 제어
    - 검색에 실패한 경우에도 모델이 무리하게 응답하면 잘못된 정보가 생성될 수 있음
    - HyDE 등의 쿼리 수정 도구를 적용하거나 컨텍스트가 없으면 응답하지 않도록 프롬프트를 설계함

RAG의 미해결 과제와 미래 전망

- 성능 저하 요인 (2/2)
  - 모델과 도메인
    - 복잡한 질의는 단순 검색만으로 해결할 수 없으며 LLM 자체의 추론 능력에 영향을 받음
    - 범용 모델은 지나치게 전문적인 도메인에서 성능이 저하될 수 있으므로 임베딩 모델과 LLM을 도메인에 맞게 파인튜닝함
  - 학습 목표
    - 임베딩 모델은 검색, LLM은 생성에 맞춰 학습되어 두 모델의 목표가 서로 다를 수 있음
    - 검색과 생성을 엔드 투 엔드로 함께 최적화하는 방식이 복잡한 질의와 전문 도메인에서 유용할 수 있음

RAG의 미해결 과제와 미래 전망

- 미래 연구 방향 (1/2)
  - 추론 고도화
    - 강화학습을 활용하여 복잡한 질의응답과 추론 성능을 개선하는 연구가 진행되고 있음
    - 지식 그래프와 그래프 기반 검색을 결합하여 문서 간 관계를 활용하려는 시도도 확대되고 있음
  - 동적 정보 통합
    - 내부 데이터베이스의 하이브리드 검색과 인터넷 검색을 결합하여 최신 컨텍스트를 확보하는 방식이 연구되고 있음
    - 데이터베이스 갱신 여부, 검색 결과 필터링, 외부 정보의 신뢰성과 보안 문제를 함께 해결해야 함

RAG의 미해결 과제와 미래 전망

- 미래 연구 방향 (2/2)
  - 전문 영역 확장
    - 수학, 의학, 생물학 등 특정 분야에 최적화된 RAG 시스템을 개발하려는 연구가 증가하고 있음
    - 도메인별 데이터와 추론 방식을 반영하면 범용 RAG보다 전문적인 질의에 효과적으로 대응할 수 있음
  - 지속되는 과제
    - RAG는 할루시네이션을 줄이지만 완전히 제거하지는 못함
    - LC-LLM, 멀티모달 데이터, 지식 그래프와의 통합에서도 성능·비용·보안의 균형이 필요함
  - 이를 통해 미래의 RAG는 단순한 문서 검색을 넘어 긴 컨텍스트, 멀티모달 정보, 지식 그래프, 실시간 인터넷 정보를 통합하는 방향으로 발전할 것으로 전망됨

Thanks!

손 우 영 (wooyoung@pel.sejong.ac.kr)