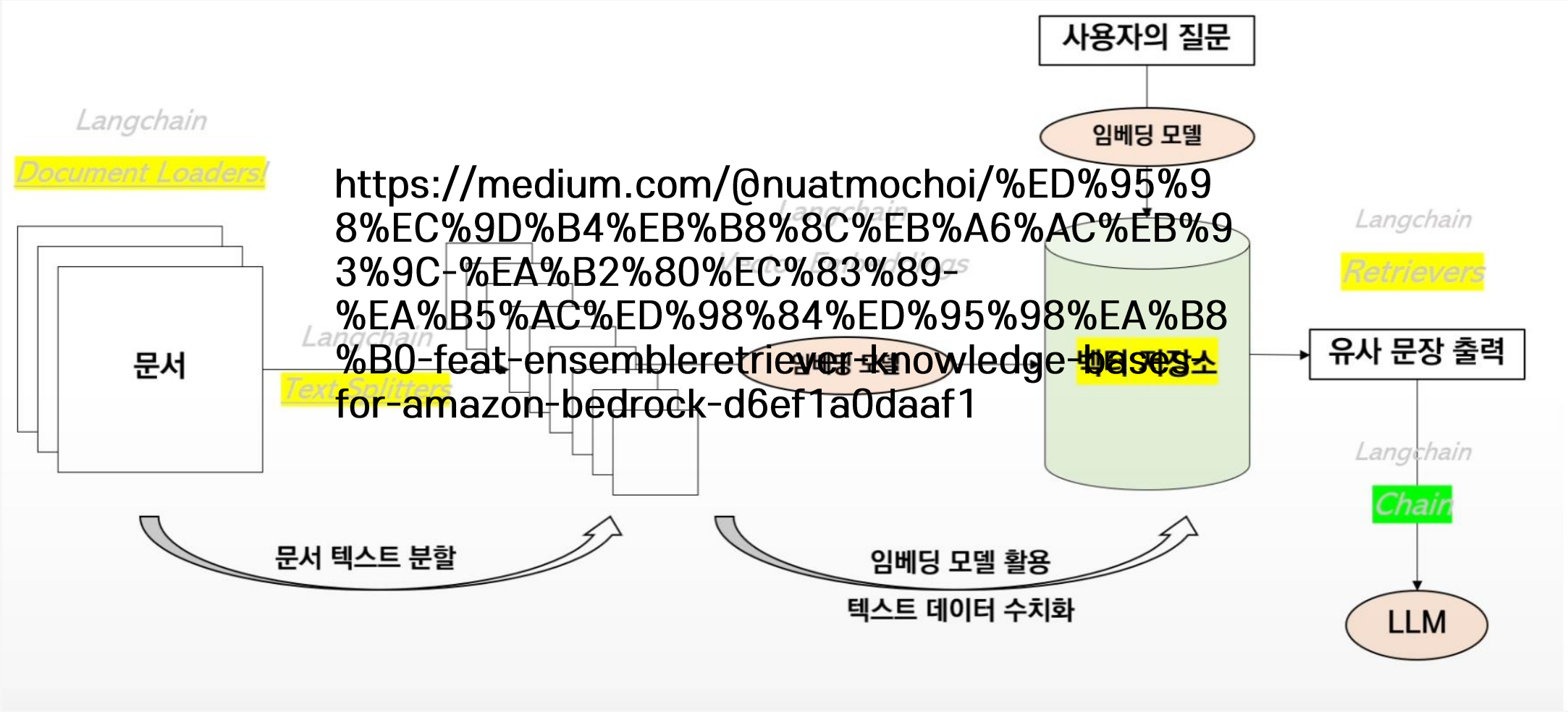
(11주차) Advanced RAG

명지대학교
MYONGJI UNIVERSITY

Vector Store 개요

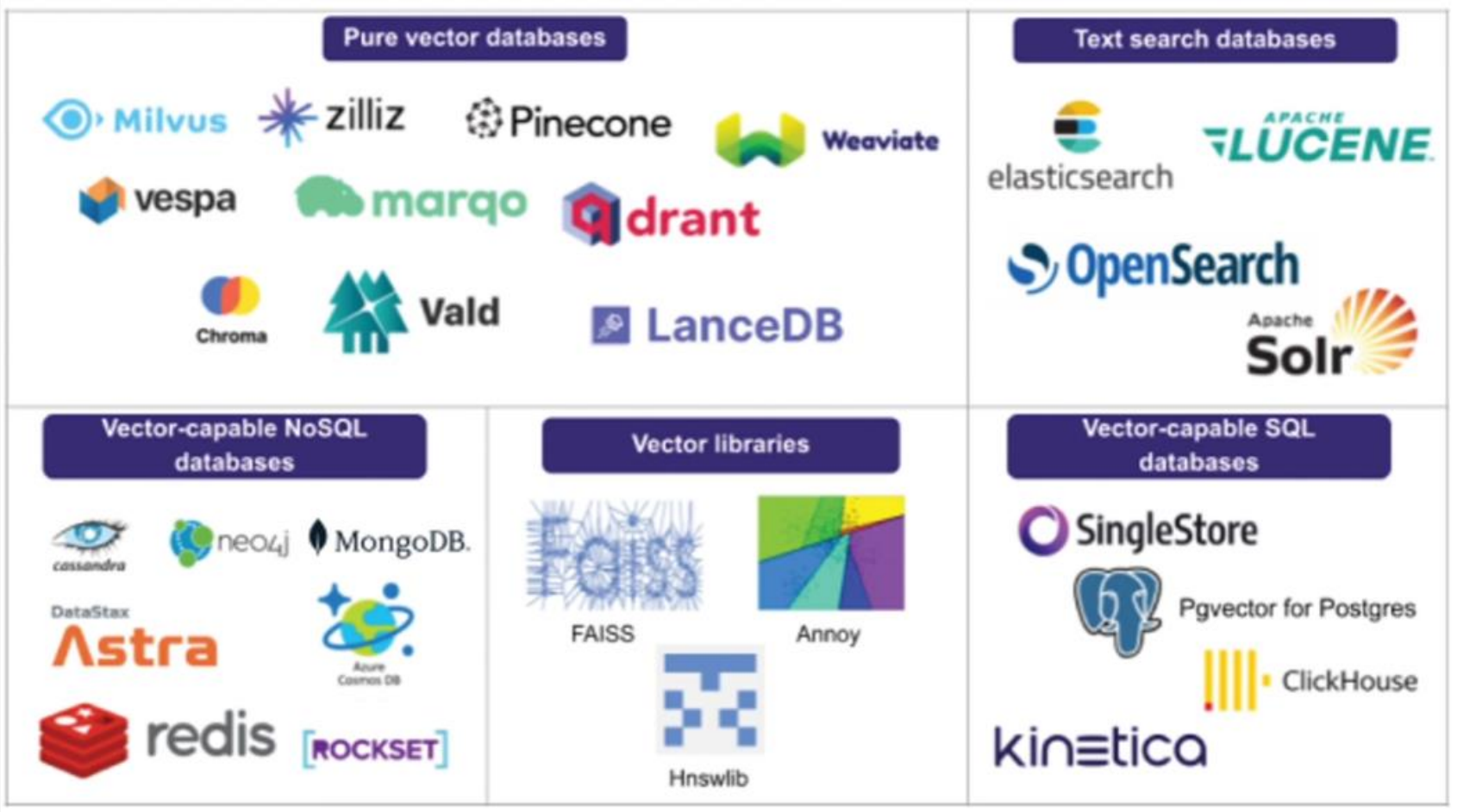
RAG Vector DB 개요



https://medium.com/@nuatmochoi/%ED%95%98%EC%9D%B4%EB%B8%8C%EB%A6%AC%EB%93%9C-%EA%B2%80%EC%83%89-%EA%B5%AC%ED%98%84%ED%95%98%EA%B8%B0-feat-ensembleretriever-knowledge-bases-for-amazon-bedrock-d6ef1a0daaf1

RAG Vector DB개요

다양한 유형과 종류의 Vector DB가 존재함



Vector Store 개요

RAG Vector DB 비교(1)

Vector Store	주요 특징/아키텍처	배포	확장성
Pinecone	관리형 서비스, 높은 확장성 및 성능, 간단한 API	관리형	매우 높음
Weaviate	오픈소스/관리형, GraphQL 인터페이스, 내장 모듈(텍스트 벡터화 등), 객체 지향 스토리지	자체 호스팅/관리형	높음
Milvus	오픈소스/관리형(Zilliz), 높은 확장성, 다중 인덱싱 알고리즘, 성능 중심	자체 호스팅/관리형	매우 높음
ChromaDB	오픈소스, 개발자 친화적, 간단한 API, RAG 애플리케이션에 중점	자체 호스팅	중간
Elasticsearch	성숙한 검색 엔진, 텍스트 검색 + 벡터 검색 지원, 분산형, RESTful API	자체 호스팅/관리형	높음
PGVector	PostgreSQL 확장 기능, SQL 생태계 활용, ACID 준수	자체 호스팅 (PostgreSQL)	PostgreSQL 의존

Vector Store 개요

RAG Vector DB 비교(1)

Vector Store	RAG 특화 기능/통합	장점	단점
Pinecone	엔터프라이즈 SLA, AI/ML 애플리케이션 최적화	운영 부담 적음, 안정적 성능	비용, 제한된 구성 옵션
Weaviate	Hugging Face, OpenAI 등 통합, 지식 그래프 기능	유연성, 모듈성, 강력한 하이브리드 검색	상대적으로 새로운 기술, 복잡성
Milvus	대규모 벡터 데이터 처리, Attu (관리 도구)	대규모 데이터셋 처리 능력 다양한 인덱스 옵션	운영 복잡성, 설정 어려움
ChromaDB	LangChain 등 RAG 프레임워크와 긴밀한 통합	사용 용이성, 빠른 프로토타이핑	대규모 운영에는 부적합 엔터프라이즈 기능 부족
Elasticsearch	고급 텍스트 분석, 집계 기능	강력한 텍스트 검색, 성숙한 기술 다양한 기능	벡터 검색 전문성은 상대적으로 낮음
PGVector	기존 PostgreSQL 데이터베이스와 통합 용이	관계형 데이터와 벡터 데이터 통합 관리 PostgreSQL의 강력한 기능 활용	순수 벡터 DB 대비 성능 제한 가능성

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG

Advanced RAG는 단순 RAG의 약점을 극복하기 위해 특정 단계를 강화

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 사전 검색 강화 (Pre-Retrieval Enhancements)

사전 검색 강화 (Pre-Retrieval Enhancements)

- 검색 단계의 입력 품질 개선, 명확한 질의를 통한 검색 효과 증대
- **인덱싱 최적화**: Chunking 최적화 , 인덱스 구조 최적화, 필터링을 위한 메타데이터 추가 정렬 최적화, 혼합 검색(mixed retrieval) 등의 전략을 포함
- **질의 최적화**: 질의 재작성, 확장 또는 변환 등을 통한 질의 명확화, 가상 문서 임베딩 (Hypothetical Document Embeddings, HyDE) 기법

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - Chunking 최적화

Chunking 최적화

- **재귀적 분할 (Recursive Chunking)**: 문단, 문장, 단어 등 의미론적 경계를 기준으로 계층적으로 분할하여 문맥 유지 노력
- **의미론적 분할 (Semantic Chunking)**: 임베딩을 사용하여 의미적으로 유사한 텍스트 블록을 하나의 청크로 묶어 문맥적 관련성을 높임
- **문서 계층 구조 활용 (Parent Document Retrieval)**: 큰 문서의 경우 요약 정보나 상위 문맥을 함께 인덱싱하거나(Parent Document Retrieval), 문서를 계층적 트리 구조로 구성하여 검색 시 더 넓은 문맥을 참고할 수 있도록 함

* Chunk 크기가 너무 크면 노이즈가 많아지고, 너무 작으면 필요한 정보가 누락될 수 있음
  중첩(Overlap)을 두어 경계에서의 문맥 손실을 줄이기도 함

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 인덱스 구조 최적화 (Optimizing Index Structures)

인덱스 구조 최적화 (Optimizing Index Structures)

검색 효율성과 정확도를 높이기 위해 인덱스 구조 자체를 개선.

- 다중 벡터 인덱싱 (Multi-Vector Indexing): 단일 문서에 대해 여러 개의 벡터 표현(예: 요약, 다른 관점의 임베딩 등)을 생성하여 검색 정확도 향상
- 그래프 인덱싱 (Graph Indexing / Knowledge Graph): 정보를 노드와 엣지로 구성된 그래프 형태로 저장하여, 의미적 관계와 구조적 정보를 함께 활용해 더 정교한 검색 수행

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 메타데이터 추가 & 정렬 최적화 (Alignment Optimization)

메타데이터 추가	각 데이터 청크에 생성 날짜, 출처, 카테고리, 작성자, 키워드 등의 메타데이터를 추가 검색 시 메타데이터를 활용하여 필터링을 수행함으로써(예: 특정 기간 내 문서, 특정 출처의 문서만 검색) 검색 범위를 축소하고 결과의 관련성을 높임 최신 자료 우선 검색 등 가중치 부여에도 활용 가능
정렬 최적화 (Alignment Optimization)	검색된 데이터가 사용자의 질의와 실제로 관련이 있는지 확인하는 과정 임베딩 모델 자체의 정렬(alignment)을 개선하거나, 검색 결과에 대해 관련성 평가 모델을 적용하는 방식 포함 가능

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 질의 최적화 (Query Optimization)

질의 최적화 (Query Optimization)

사용자로부터 입력받은 원래의 질의(Query)를 검색에 더 효과적인 형태로 가공하거나 변환

- 질의 재작성 (Query Rewriting): 사용자의 원래 질의가 모호하거나 너무 짧을 경우, LLM 등을 사용하여 더 명확하거나 상세한 질의로 재작성, 여러 관점에서 질의를 생성하여 다중 검색 후 결과를 통합하는 방식(Multi-query rewriting)도 포함
- 질의 확장 (Query Expansion): 원래 질의에 동의어나 관련 키워드를 추가하여 검색 범위를 넓히고 관련 문서를 더 많이 찾을 수 있도록 함

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 질의 최적화 (Query Optimization)

질의 최적화 (Query Optimization)	- 질의 변환 (Query Transformation): 복잡한 질의를 더 간단한 하위 질의들로 분해(Problem decomposition) 스텝-백 프롬프팅(Step-Back Prompting): 질의에서 핵심 개념을 추출하여 더 추상적이거나 일반적인 질문으로 변환한 후, 이를 바탕으로 넓은 범위에서 검색하고 다시 원래 질의와 연결 - 가상 문서 임베딩 (HyDE - Hypothetical Document Embeddings): 사용자의 질의에 대한 이상적인 답변(가상 문서)을 LLM으로 먼저 생성 이 가상 문서를 임베딩하여 실제 문서 코퍼스에서 유사한 문서를 검색 질문 자체보다 질문에 대한 (가상의) 답변이 실제 관련 문서와 의미적으로 더 유사할 수 있다는 가정에 기반

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 검색 단계 개선(Retrieval Stage Improvement)

검색 단계 개선 (Retrieval Stage Improvement)

핵심 검색 메커니즘을 강화하는 데 초점

- 임베딩 모델 미세 조정(Fine-tuning Embedding Models): 특정 도메인에 임베딩 모델을 적응시켜 관련성을 높임

  사전 학습된(pre-trained) 임베딩 모델은 방대한 일반 텍스트 데이터로 학습되어 다양한 주제에 대해 우수한 성능을 보장하지만 특정 전문 분야나 특정 기업/기관 내부에서만 사용되는 용어, 약어, 개념 등에 대해서는 일반적인 의미와 다르게 해석되거나, 모델이 해당 용어의 특수한 문맥적 의미를 충분히 파악하지 못할 수 있음

  이러한 문제를 해결하기 위해, 기존의 사전 학습된 임베딩 모델을 특정 도메인(예: 법률, 의료, 금융이나 특정 회사의 내부 문서 등)의 데이터셋을 사용하여 파인 튜닝 후 임베딩 모델로 활용

- 하이브리드 검색(Hybrid Search): 의미론적(벡터) 검색과 키워드 기반 검색을 결합

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 사후 검색 처리(Post-Retrieval Processing)

검색 단계 개선 (Retrieval Stage Improvement)

생성 전에 검색된 정보를 정제하는 데 중점

- **재순위화(Re-ranking)**: 교차 인코더(cross-encoder)와 같은 더 정교한 모델을 사용하여 초기 검색된 Chunk들의 순위를 재조정하여 가장 관련성 높은 순서대로 재 순위화
- **컨텍스트 압축/선택(Context Compression/Selection)**: 노이즈나 관련 없는 정보를 필터링, 검색된 컨텍스트를 요약하거나 압축하여 Context Window 크기에 맞춤

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 사후 검색 처리(Post-Retrieval Processing)

교차 인코더 (Cross-Encoder)를 이용한 재순위화

재순위화에는 다양한 모델과 기법이 사용될 수 있는데, 그중에서도 교차 인코더(Cross-Encoder)는 높은 정밀도를 제공하는 강력한 방법

- 입력 방식: 교차 인코더는 사용자의 질의(Query)와 검색된 각 문서 청크(Document Chunk)를 하나의 쌍(pair)으로 묶어 동시에 입력으로 활용
- 상호 작용 분석: 입력된 질의-문서 쌍 내부에서 단어 간, 구문 간의 복잡한 상호작용(interaction)과 문맥적 관계를 심층적으로 분석
  BERT와 같은 트랜스포머(Transformer) 기반의 대규모 언어 모델 아키텍처를 활용
- 관련성 점수 출력: 각 질의-문서 쌍에 대해 얼마나 관련성이 있는지를 나타내는 단일 점수(relevance score)를 출력(예, 0과 1 사이의 값으로 표현)
- 재순위화: 관련성 점수가 높은 순서대로 초기 검색된 문서 청크들의 목록을 재순위화

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 사후 검색 처리(Post-Retrieval Processing)

교차 인코더 (Cross-Encoder)를 이용한 재순위화

- Bi-Encoder (일반적인 벡터 검색에서 사용): 질의와 문서를 각각 독립적으로 인코딩하여 임베딩 벡터를 생성한 후, 두 벡터 간의 유사도(예: 코사인 유사도)를 계산
  계산 효율성이 높아 대량의 문서에 대한 초기 검색에 적합하지만 질의와 문서 간의 직접적인 상호작용을 모델링하지 않기 때문에 미묘한 문맥적 관련성 누락 가능
- Cross-Encoder: 질의와 문서를 함께 입력받아 상호작용을 직접 모델링하기 때문에 Bi-Encoder보다 훨씬 더 정교하고 정확한 관련성 판단이 가능
  높은 정밀도: 질의와 문서 간의 심층적인 의미론적 관계를 평가하여 매우 정확한 관련성 점수를 제공
  문맥 이해 능력 우수: 단순 키워드 매칭이나 표면적인 유사성을 넘어, 복잡한 문맥과 뉘앙스를 파악하는 데 강점

RAG(Retrieval-Augmented Generation) 개선

Advanced RAG - 사후 검색 처리(Post-Retrieval Processing)

교차 인코더 (Cross-Encoder)를 이용한 재순위화

- Cross-Encoder는 각 질의에 대해 검색된 모든 후보 문서와 쌍을 이루어 개별적으로 모델에 입력으로 제공해야하므로, 계산량 과다

수만 개의 문서 전체에 대해 교차 인코더를 직접 사용하는 것은 비효율적

이러한 계산 비용 문제를 해결하기 위해 일반적으로 두 단계 검색 전략을 사용

1단계 (초기 검색/후보군 생성): Bi-Encoder를 사용한 벡터 유사도 검색 방법으로 상위 K개의 후보 문서 Chunk를 추출

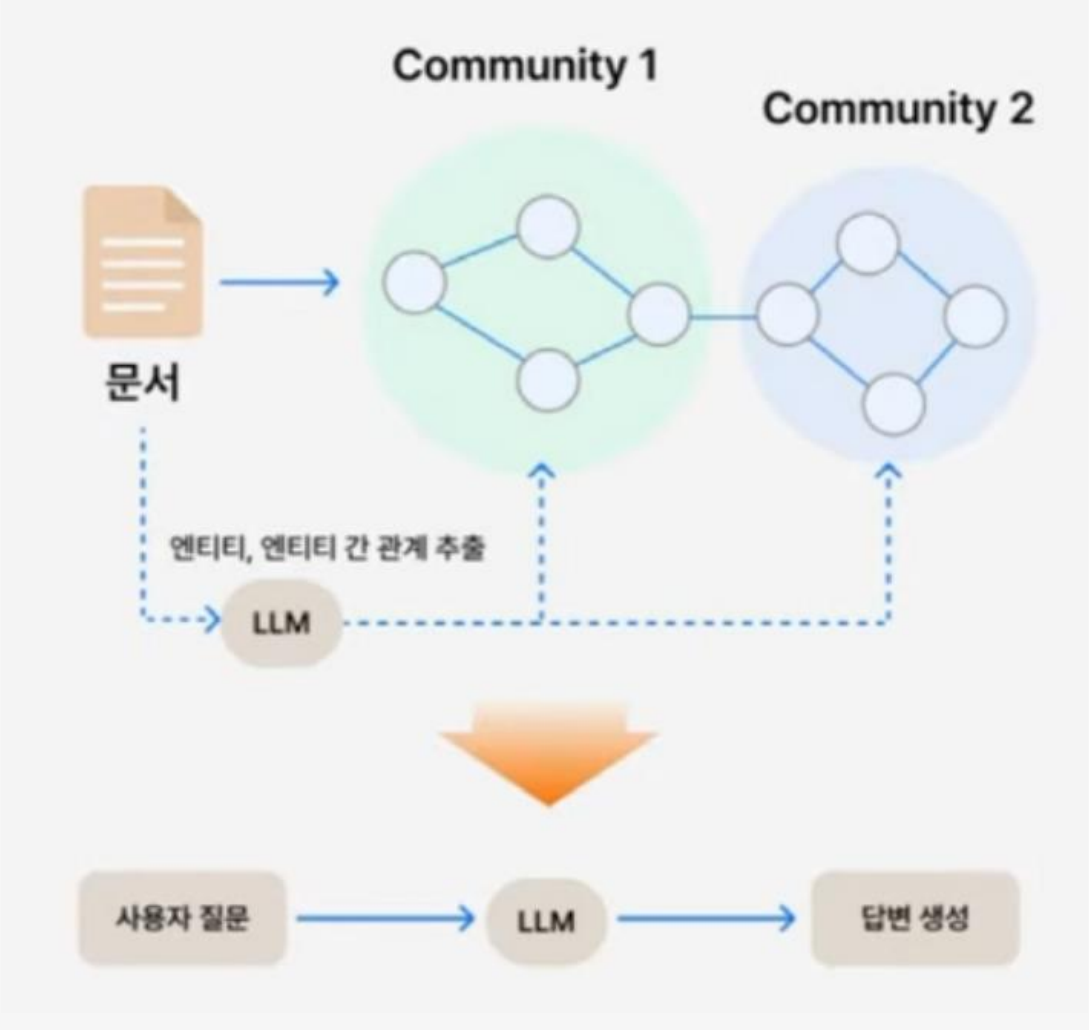
2단계 (재순위화): 1단계에서 추출된 비교적 적은 수의 후보 문서 청크들(예: 상위 64개)에 대해서만 교차 인코더를 적용하여 정밀하게 재순위화

이를 통해 전체적인 효율성을 유지하면서 최종 결과의 정밀도 향상 가능

RAG(Retrieval-Augmented Generation) 트렌드

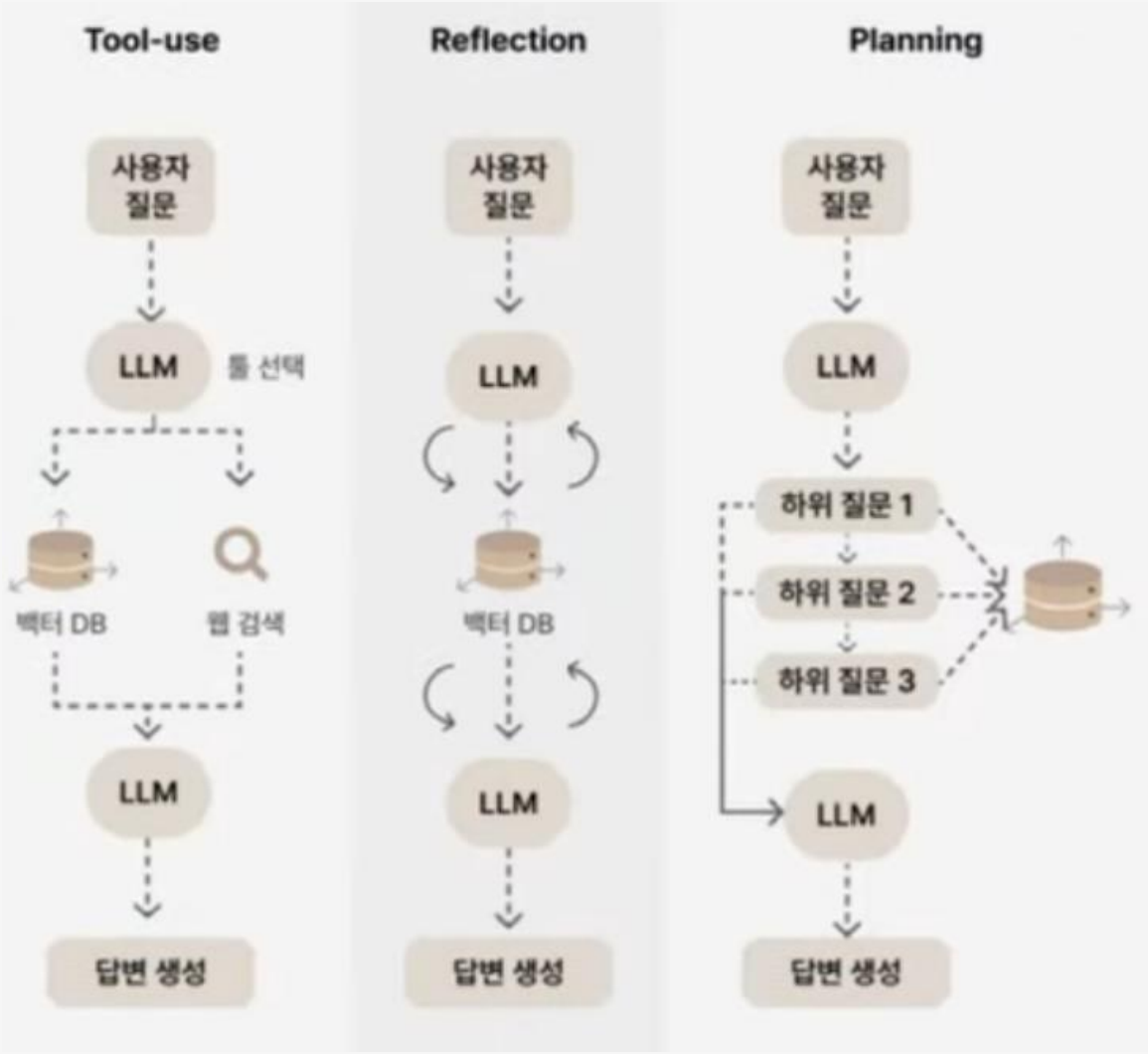
Graph RAG

지식 그래프(Knowledge Graph)의 구조적인 이점을 결합하여, 정보 검색의 정확도와 생성되는 답변의 문맥적 이해도를 한층 높인 발전된 형태의 RAG 접근법



RAG(Retrieval-Augmented Generation) 트렌드

Agentic RAG



RAG(Retrieval-Augmented Generation) 트렌드

Agentic RAG

Agentic RAG

- 개념: LLM이 지능형 에이전트처럼 자율적으로 행동, 문제 분석, 계획 수립, 동적 정보 검색, 다양한 도구 활용, 정교하고 상황에 맞는 답변 제공
  단순 "검색 후 생성"을 넘어 **반복적 추론, 도구 사용, 자기 평가 및 수정 과정** 포함
- 동작 방식
  1. **지능형 에이전트 (Intelligent Agents)**: 하나 이상의 LLM 기반 에이전트가 핵심 목표 달성을 위해 자율적 행동, 복잡한 질문/작업을 하위 작업으로 분해, 필요한 정보/도구 결정
  2. **동적이고 반복적인 정보 검색 및 처리**: 정적 파이프라인 대신 상황에 따라 검색 전략 실시간 조정, 필요시 쿼리 수정, 다중 데이터 소스 접근, 외부 도구(예: 웹 검색) 사용 만족스러운 결과 얻을 때까지 반복 가능

RAG(Retrieval-Augmented Generation) 트렌드

Agentic RAG

Agentic RAG

3. 도구 사용 (Tool Utilization): 다양한 "도구" 사용 가능 (예: DB 쿼리 실행기, 코드 실행기, API 호출, 웹 브라우저), LLM 단독 수행 어려운 작업 처리 (실시간 정보 접근, 외부 서비스 연동)
4. 계획 수립 및 추론 (Planning and Reasoning): 목표 달성 위한 계획 수립, 수집 정보 기반으로 추론하여 다음 행동 결정, ReAct, Chain-of-Thought 등 다양한 추론 기법 활용 가능
5. 자기 평가 및 피드백 루프 (Self-Correction and Feedback Loop): 에이전트가 검색 정보 관련성, 생성 답변 품질 등 스스로 평가, 필요시 쿼리 수정, 추가 정보 검색 등 개선 작업 반복 수행, 지속적인 학습 및 개선 유도
6. 메모리 활용 (Memory Utilization): 이전 대화, 수행 작업, 획득 정보 등을 "메모리"에 저장 및 활용, 시맨틱 캐싱 등으로 유사 쿼리/결과 재활용하여 효율성 증대

Agentic RAG의 Agent 유형

라우팅 에이전트

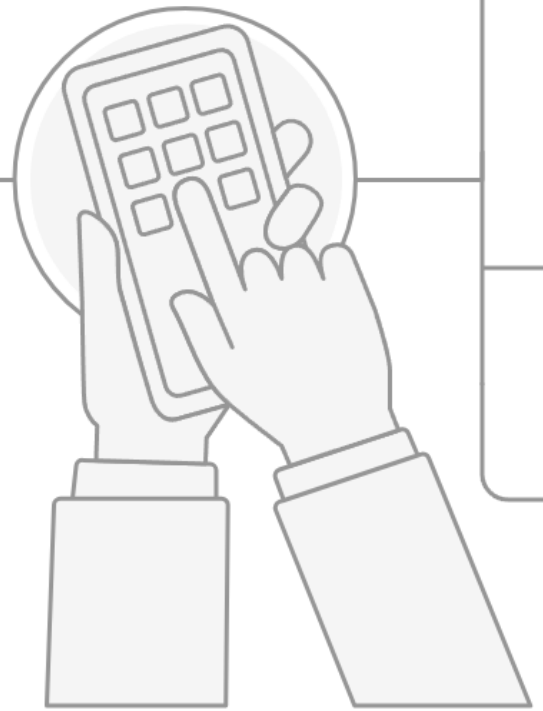
사용자 질의를 분석하고 적절한 에이전트, 도구 또는 파이프라인을 선택

검색 에이전트

다양한 데이터 소스에서 정보를 검색

컨텍스트 조합/평가 에이전트

수집된 정보를 조합하고 관련성/신뢰도를 평가하여 최종 컨텍스트를 구성



쿼리 계획/수정 에이전트

복잡한 질의를 분해하거나 재작성

도구 사용 에이전트

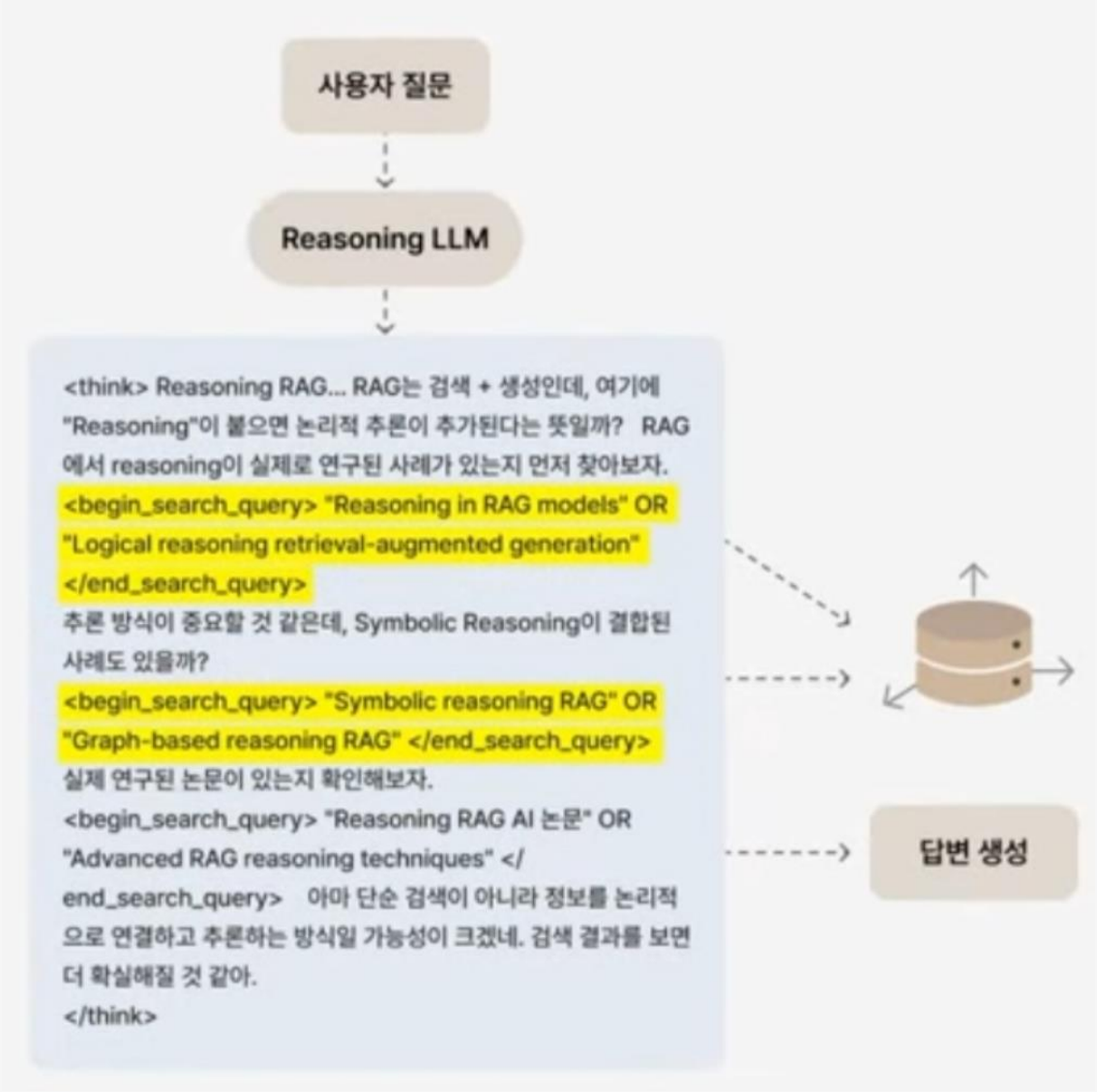
필요시 외부 도구를 활용

응답 생성 에이전트

최종 컨텍스트와 원본 질의를 기반으로 답변 생성

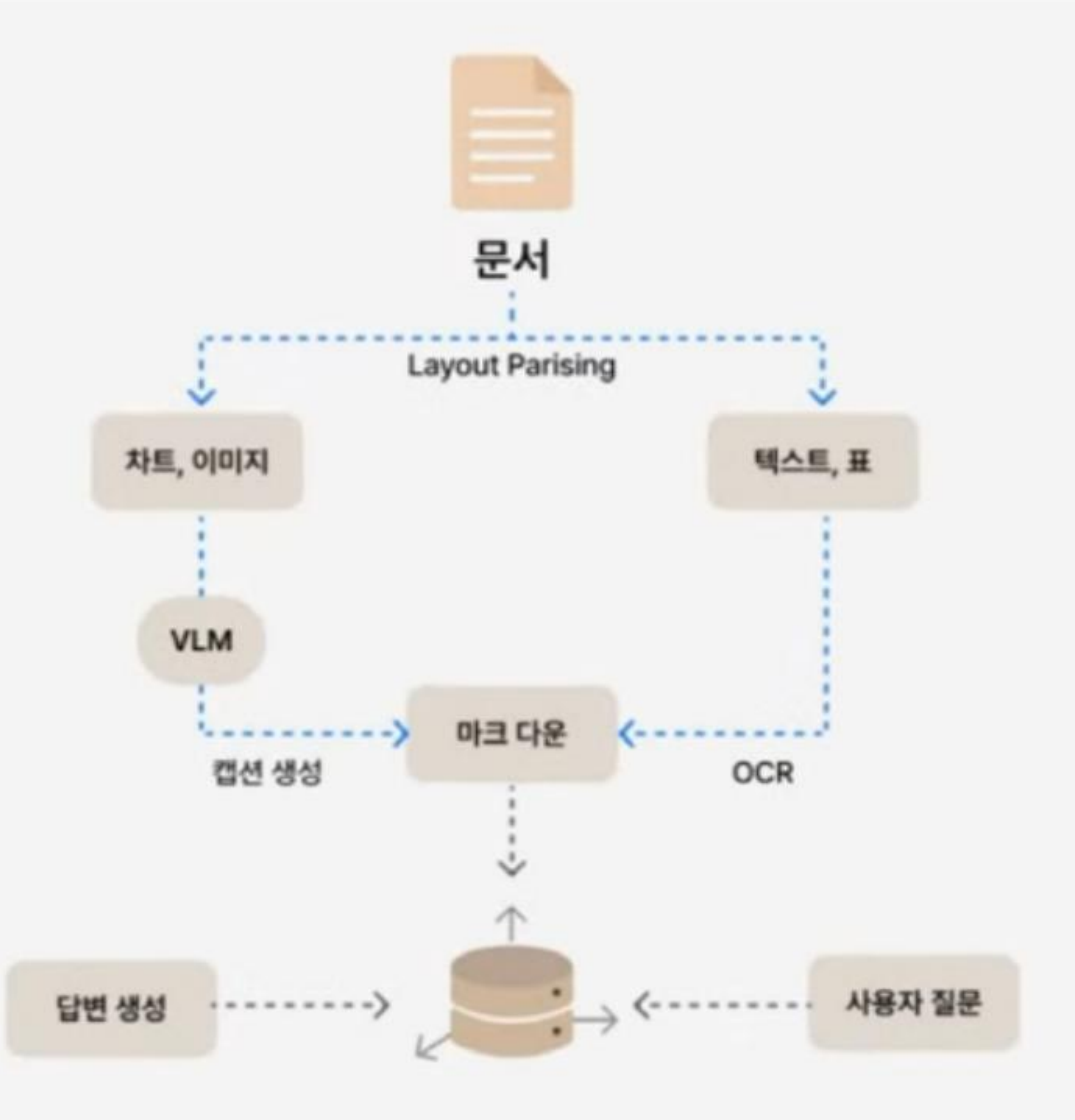
RAG(Retrieval-Augmented Generation) 트렌드

Reasoning RAG



RAG(Retrieval-Augmented Generation) 트렌드

Multi-Modal RAG



RAG(Retrieval-Augmented Generation) 트렌드

Multi-Indexing RAG