RAG 기술 조사

참고논문 : Retrieval-Augmented Generation for Large Language Models: A Survey

인공지능전공

K2024026 황현준

목차

01. RAG란?

RAG 뜻

- Retrieval-Augmented Generation
- 검색(회수)-증강 생성
- 영어권 발음은 '랙' 한국은 '레그'로 읽음

단어 풀이

❖ Retrieval

- 사용자가 궁금해하는 질문에 대한 답을 가지고 있을 법한 정보를 찾아서 가져오고

❖ Augmented

- 그 정보를 중요하게 여기게 하고 (= 가져온 정보로 양을 늘림)

❖ Generation

- 답을 생성하자

01. RAG란?

다양한 출처에서의...

AWS	대규모 언어 모델의 출력을 최적화하여 응답을 생성하기 전에 학습 데이터 소스 외부의 신뢰할 수 있는 지식 베이스를 참조하도록 하는 프로세스
Elastic	프라이빗 또는 독점 데이터 소스의 정보로 텍스트 생성을 보완하는 기술
LangChain	정보를 가져와 모델 프롬프트에 삽입하는 프로세스

외부 지식 베이스를 참조하여 LLM이 원하는 답변을 생성하도록 하는 기술

02. RAG 탄생배경(fit 기존 한계점)

❖ LLM만을 사용 했을 때의 문제(=한계점)

신뢰할 수 없음

- 신뢰할 수 없는 출처로부터 응답을 생성
- hallucination(환각) 현상 발생

원하는 응답 생성 어려움

- 구체적이고 최신의 응답을 기대할 때 오래되었거나 일반적인 정보를 제공
- 용어 혼동(동음이의어)으로 인해 응답의 정확도 저하

지식의 업데이트 or 수정이 어려움

- 암시적으로 모델 network에 지식이 저장되어 있고 그렇기 때문에 외부 데이터에 대한 접근 X
- 최신 정보 추가가 어렵다
- 어느 위치에 지식이 저장되는지 모르기 때문에 잘못된 정보가 있어도 수정하기가 어려움
- 모델의 크기가 지식의 저장 가능한 최대 크기에 제한이 됨

02. RAG 탄생배경(fit 기존 한계점)

❖ 한계 극복을 위한 방향성

1. Fine-tuning

- 도메인 특화 데이터로 모델을 추가 학습
- Instruction finetuning을 통해 zero-shot 성능을 높이기
- 예) RLHF 학습 (인간 피드백을 통한 강화학습)

2. Prompt Engineering

- In-context learning(ICL) 라는 개념을 기반으로 프롬프트를 수정하여 원하는 답변을 얻는 방법
- 예) one shot, few-shat, Chain-of-Thought 등등

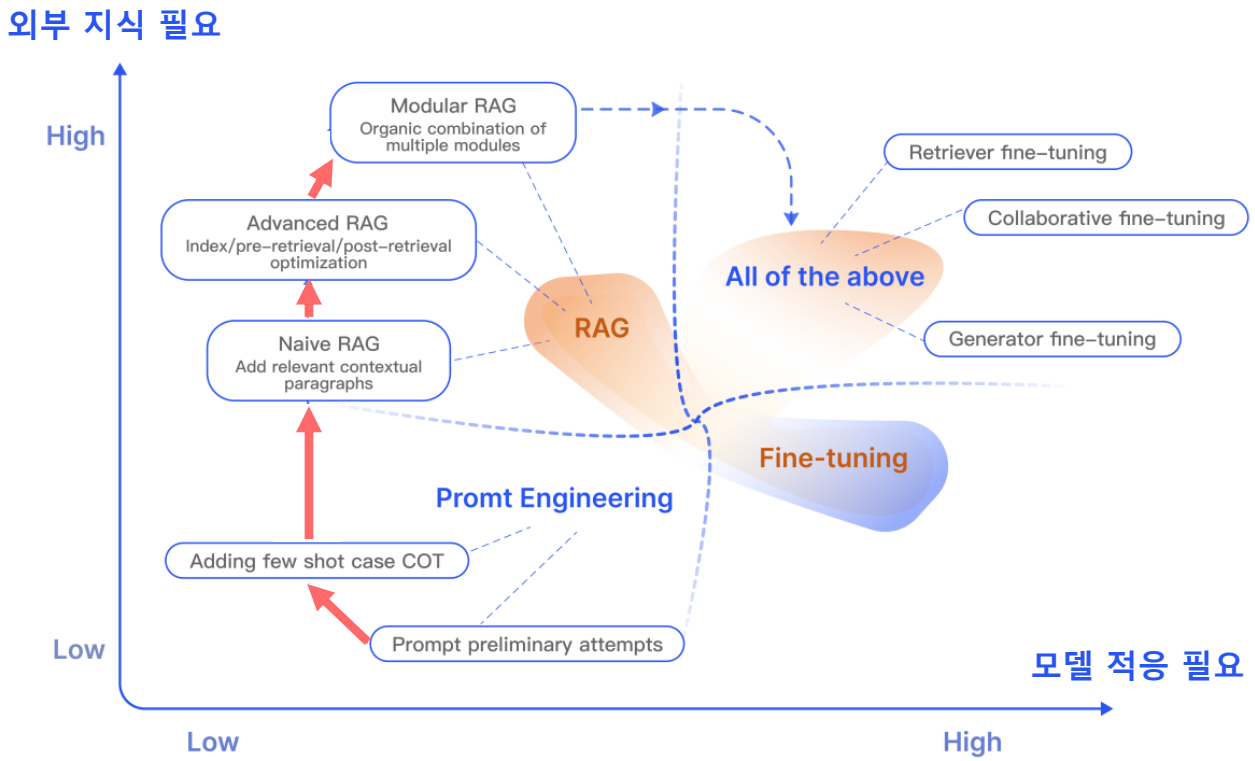


Figure 2: RAG compared with other model optimization methods

02. RAG 탄생배경(fit 장단점)

❖ RAG의 장단점

장점

- **최신정보**
  - 외부 참조를 정기적으로 지식 업데이트 가능
- **비용 효율적**
  - 파인 튜닝에 비하여 지식을 업데이트 하는 비용이 훨씬 낮다
  - 컴퓨팅의 저장공간을 더 적게 쓰고 지식 저장 가능
  - 자체 LLM을 보유할 필요가 없음(보안 요구에 따라 다름)
- **개발자 제어 강화**
  - 데이터 엑세스 제어를 통헤 문서 레벨의 보안 구현 가능
  - 잘못된 데이터의 수정 or 삭제 가능
  - 지식 저장소에 일반화된 LLM 데이터보다 맥락성이 강한 데이터를 포함 할 수 있다
- **사용자 신뢰 강화**
  - hallucination(환각)의 발생을 억제
  - 지식 집약적 작업에서 모델의 정확도를 향상 시키는데 도움이 됨
  - 답변의 출처를 표기 가능 (답변의 투명성 강화)
  - 개인에게 맞춤화된 답변 생성 가능

    ⇒ 사용자 개인정보가 추가된 답변이라던가..

    ⇒ 특정 영역에 특화된 답변이라던가...

02. RAG 탄생배경(fit 장단점)

❖ RAG의 장단점

단점

- **외부 지식에 의존**
  - 검색된 정보가 올바르지 않으면 부정확한 결과가 나올 수 있음

02. RAG 탄생배경 (fit 곁들임 질문)

❖ Context window 를 늘리면 RAG 필요 없는 거 아니야?

예시 질문)

책 한권을 통째로 LLM 에게 입력하고 질문을 해보니까, 잘 대답하는데 RAG를 사용할 필요가 있을까?

- LLM은 의미적으로 관련이 있지만 **관련성이 없는 내용으로 인해 주의가 산만**해지는 경향이 있음
- 여러 버전이 있는 문서를 모두 LLM에게 제공하고 **답변이 상충** 될 수 있음
- 수백만개의 무관한 토큰을 처리하는 일은 **많은 비용과 시간이 소요**됨
- Context 가 너무 길면 **중간 위치의 의미가 소실**되는 현상이 있음

03. RAG 패러다임 3종

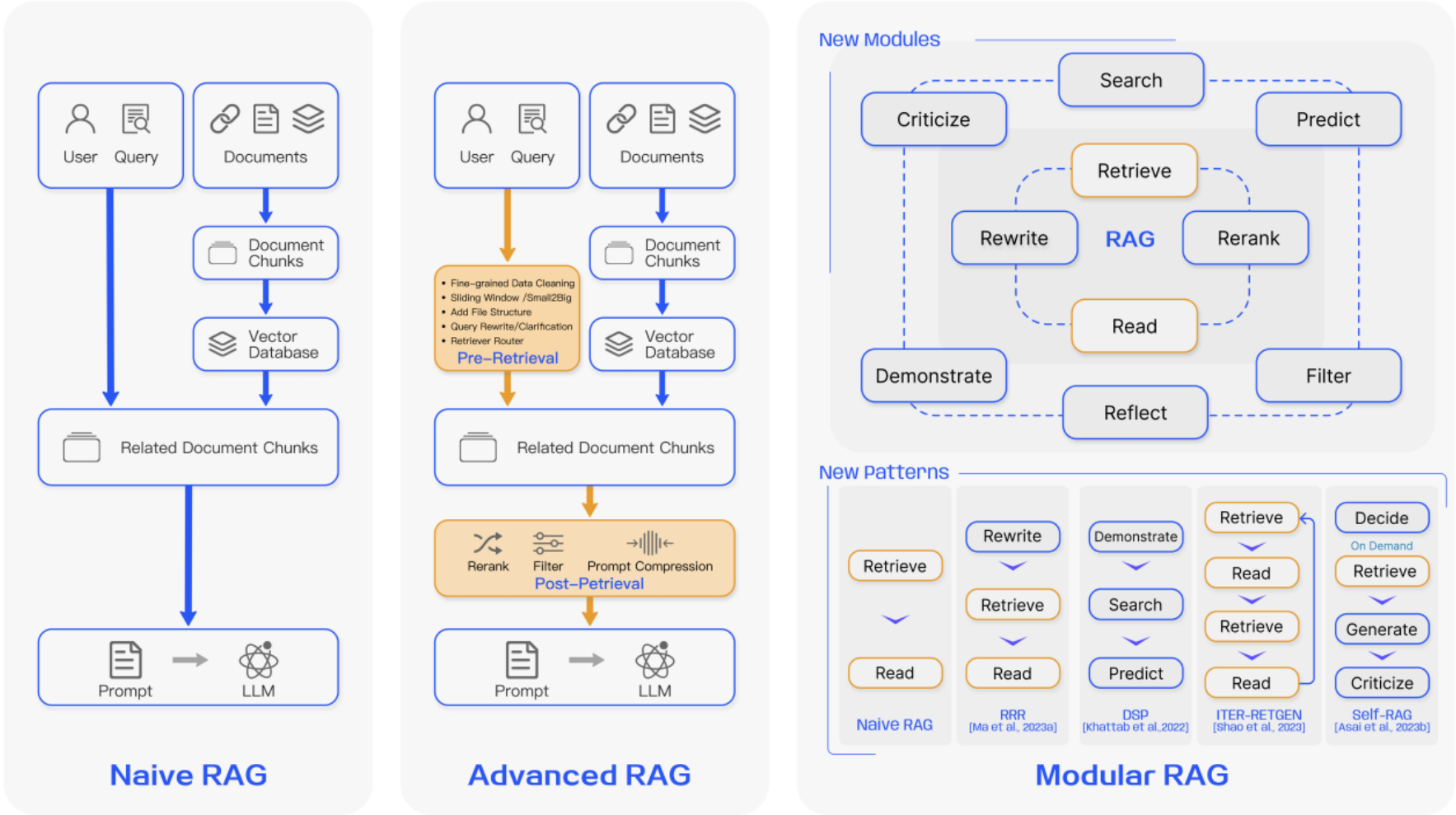
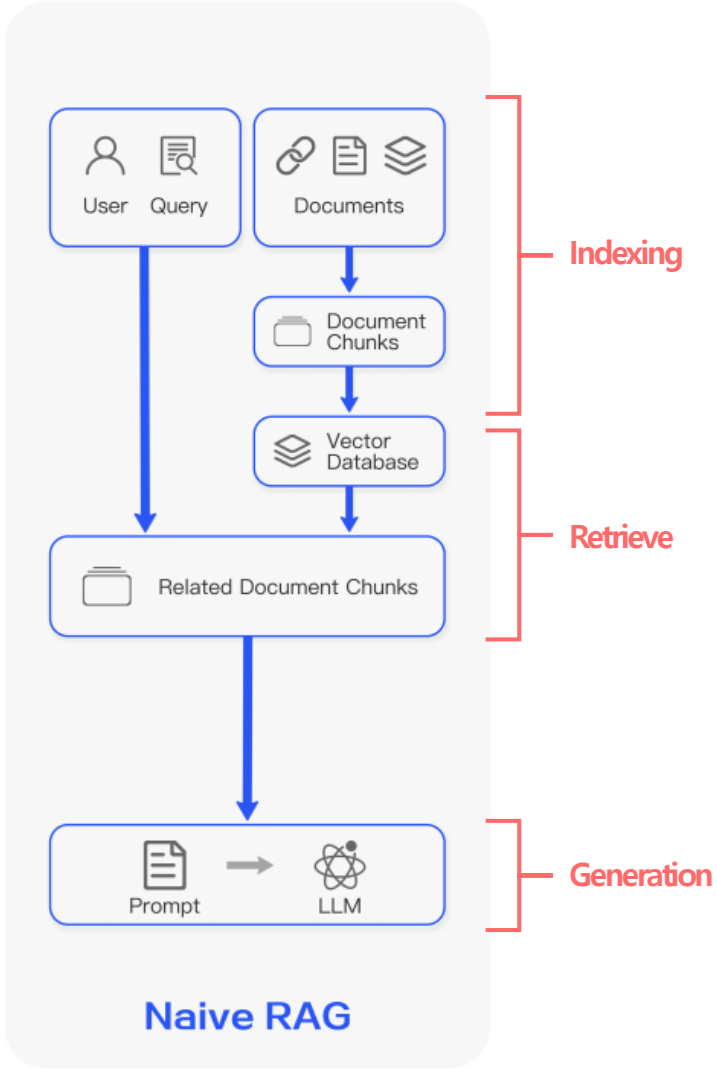


Figure 3: Comparison between the three paradigms of RAG

03. RAG 패러다임 – Naive RAG



1. Indexing

- 원본 데이터를 추출하고 블록(chunk) 단위로 나눈 후 embedding model을 사용하여 각 문서 블록의 vector를 생성

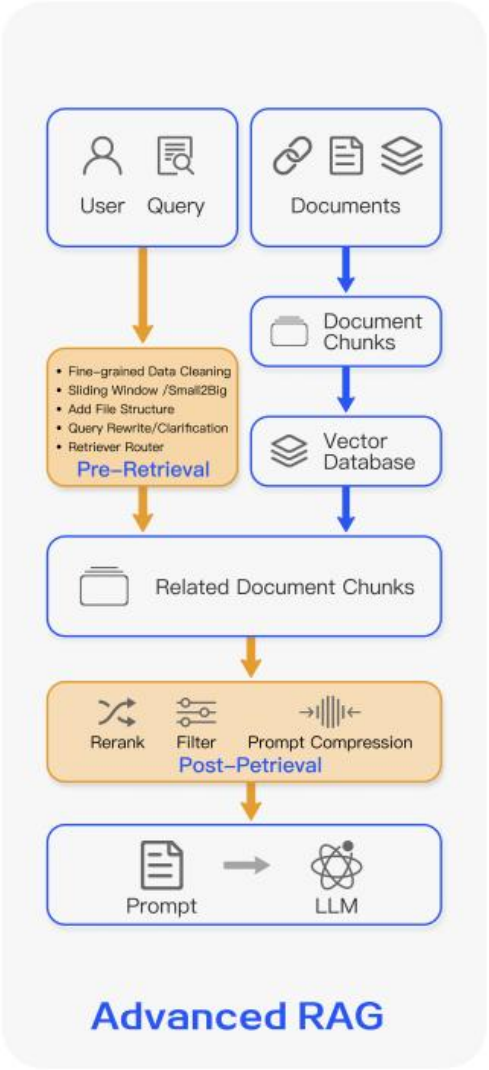
2. Retrieve - 검색

- 사용자의 입력을 vector로 변환하고 DB에서 문서 블록의 vector 간의 유사도를 계산하여 상위 블록 선택

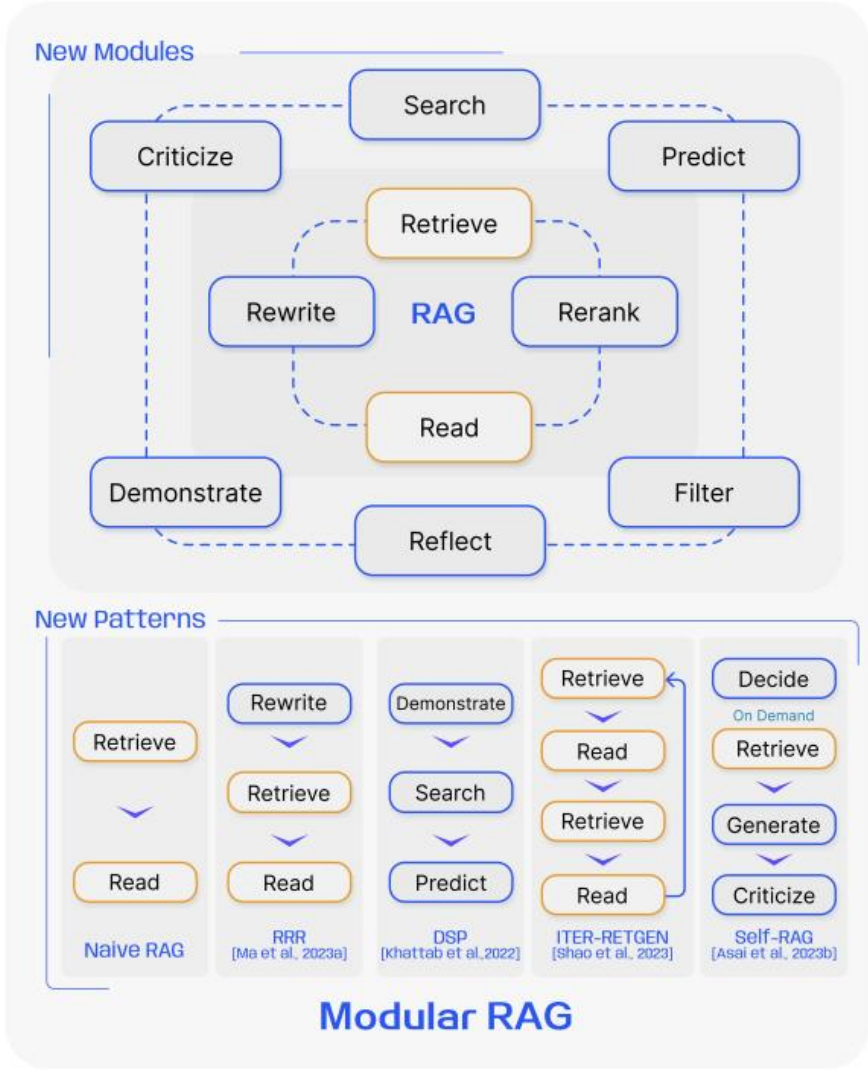
3. Generation - 생성

- In-context learning(ICL) 라는 개념을 기반으로 프롬프트를 수정하여 원하는 답변을 얻는 방법
- 예) one shot, few-shat, Chain-of-Thought 등등

03. RAG 패러다임



- Naive RAG 의 부족한 점을 개선하기 위한 패러다임
- 3가지 단계로 구분되어 개선 방법 추가
  - **검색 전 절차**
  - **검색 후 절차**
  - **RAG 파이프라인 최적화**



- 다양한 시나리오와 요구사항에 맞게 조정할 수 있도록 여러 모듈과 패턴을 추가한 패러다임
- 여러 검색 방법 들을 검색 모듈에 통합하건, 리트리버에 파인튜닝 접근 방식을 추가하는 등

다양한 방법들을 모듈 단위로 통합 및 확장

04. RAG 토이 프로젝트

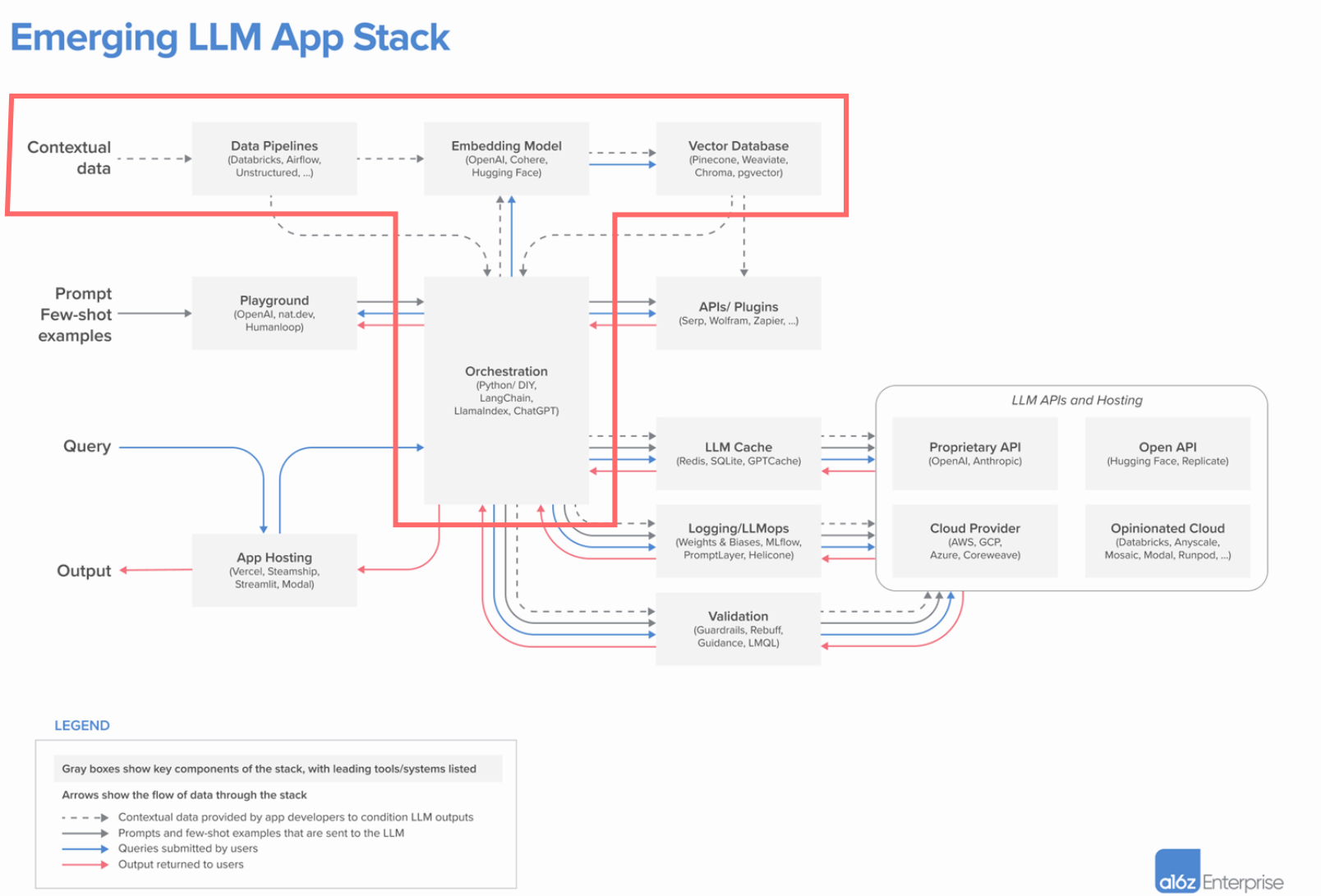
❖ 토이 프로젝트 시작

목표

- Naive RAG를 구현 및 테스트
- RAG의 최소 단위 구현을 통해 각 요소의 이해도 증대

방법

- Orchestration 프레임워크를 중 하나를 선택하여 표시영역까지 구현 및 테스트
- **LangChain** 를 선택하여 진행



05. RAG 고급 테크닉 및 전략

❖ 모듈러 RAG 프레임워크

3계층 구조

- Module Type
  - Module
    - Operators

05. RAG 고급 테크닉 및 전략

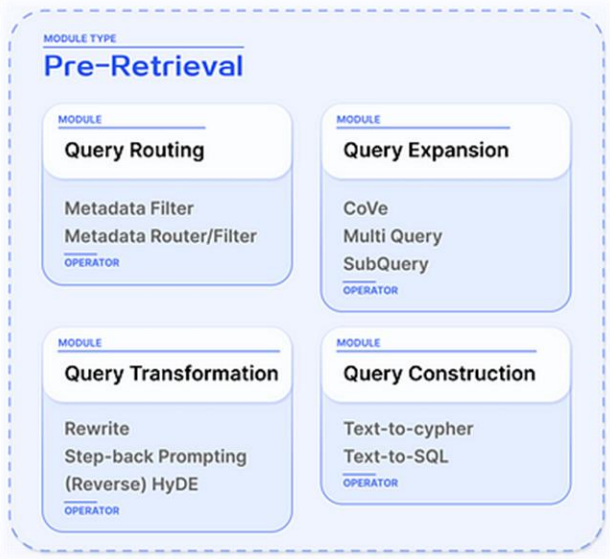
➢ Chunk Optimizaion – 청크 최적화

- **Small to Big** : 작은 청크로 검색 후 부모 청크의 ID 참조
- **Sliding window** : 청크를 겹치게 자르기
- **Summary** : 큰 청크를 요약하고, 요약으로 검색
- **Matadata Attachment** : 메타데이터를 통해 검색 필터링

➢ Structural Organization

- **계층적 인덱스** :
  데이터들의 관계를 트리 구조와 같은 계층적 관계로 만들어 관리하는 것
  문서 구조적 or 의미론적으로 구조화 시킬 수 있음
- **KG Organization** :
  지식 그래프를 활용하여 구조화시키기

05. RAG 고급 테크닉 및 전략



➢ Query expansion – 쿼리 확장

- **Multi Query** : 원본 질문보다 더 넓은 범위의 쿼리를 추가
- **SubQuery** : 원본 질문을 맥락화하고 필요한 하위 질문을 생성
- **CoVe** : 응답을 검증하는 쿼리를 생성해서, 검증된 응답을 얻는다

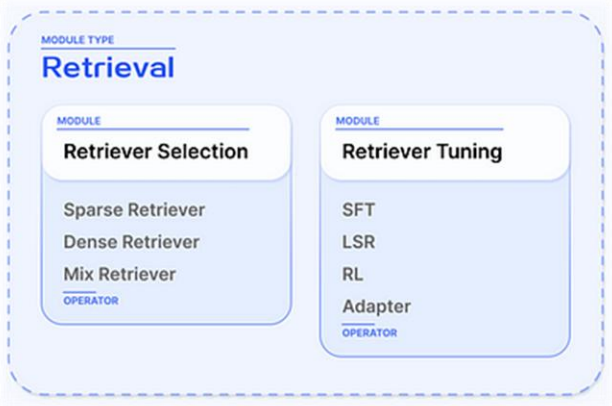
➢ Query Transformation – 쿼리 변환

- **Rewrite** : LLM이 질문을 다시 쓰기
- **HyDE** : 원본 질문의 가상 문서를 생성하고, 가상문서로 검색 (역관계 O)
- **Step-back Prompting** : 원본 질문을 추상화 하여 고수준 개념 질문 생성 후 검색

➢ Query Routing – 쿼리 라우팅

- **Metadata Filter** : 질문에서 키워드 추출하고, 키워드로 검색 필터링
- **의미 라우팅** : 특정 응답이나 동작을 트리거 하는 쿼리 목록

05. RAG 고급 테크닉 및 전략



➢ Retriever Selection – 리트리버 선택

- **Sparse Retriever** : 단어 빈도 같은 통계적 방법 기반으로 검색 수행
- **Dense Retriever** : 의미론적 유사도로 검색 수행
- **Mix Retriever** : Sparse 와 Dense 를 혼합하여 사용

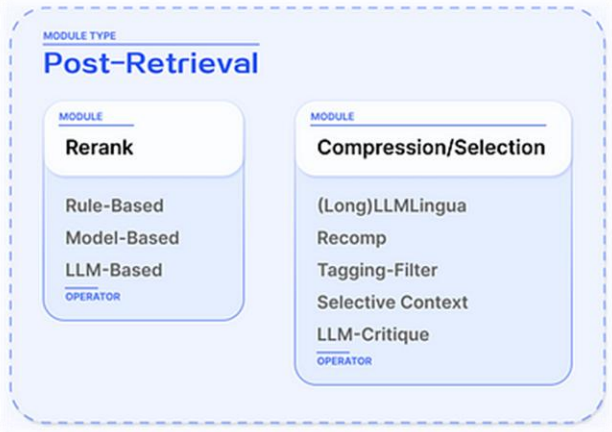
➢ Retriever Tuning

- **LSR** : LM에서 생성된 결과로 임베딩 모델 파인튜닝
- **Adapter** :

  모델의 직접적인 파인튜닝이 불가피할 때,

  모델 외부에 어댑터를 추가적으로 부착하는 방식

05. RAG 고급 테크닉 및 전략



➢ Rerank

- **Rule-Based** : 다양성, 관련성, MRR 같은 지표로 규칙에 따라 청크를 다시 정렬
- **Model-Based** : 문서 청크 재정렬 하는 LM 활용

➢ Compression/Selection

- **(Long)LLMLingua** : LM을 사용하여 프롬프트에서 중요하지 않은 토큰 제거 및 형식 변환
- **Recomp** : 문서의 관련 문장을 선택하고 통합하여 간결한 요약 생성
- **Selective Context** : 입력 컨텍스트의 중복된 내용을 식별하고 제거
- **LLM-비평** : LLM이 최종 답변을 생성하기 전에 검색된 문서를 평가

05. RAG 고급 테크닉 및 전략

➢ Scheduling

- **Rule-base** : 정의된 규칙에 따름
- **Prompt-base** : LLM이 다음 행동 방향 결정
- **Tuning-base** : LLM이 특정 작업을 트리거하는 특정 토큰 생성 (Self-RAG)

➢ Fusion

- **Possibility 앙상블** : LM을 활용하여 Retriever를 학습하는 방식
- **RRF** : 여러 검색 결과 목록의 순위를 결합하여 단일 통합 순위를 생성하는 기술

감사합니다