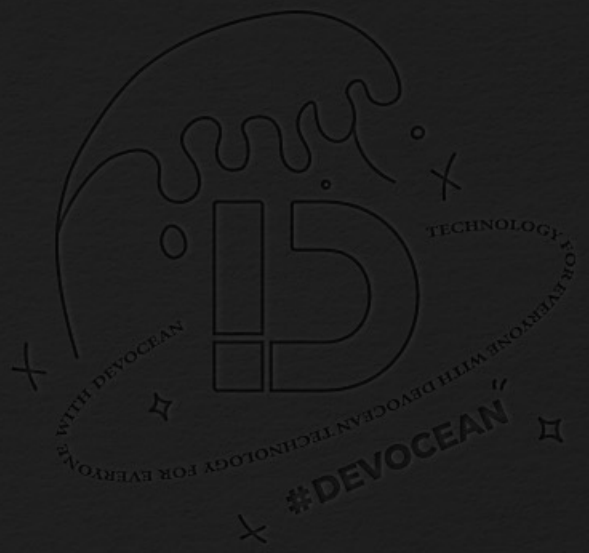
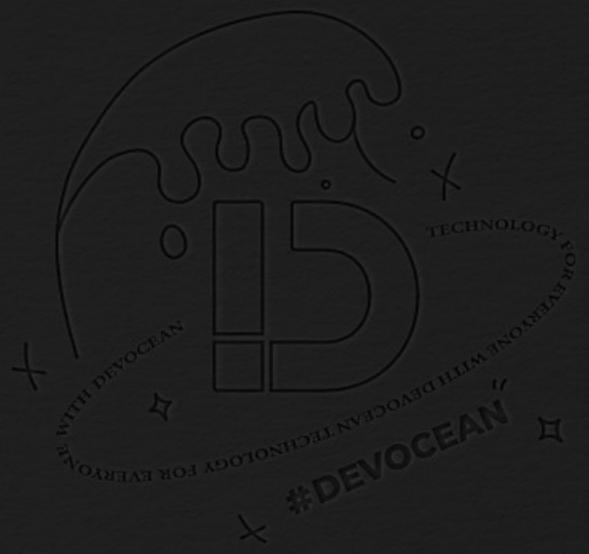
DEVOCEAN

OpenLLM / RAG 업무 필요 사례

현대오토에버 이길호



목차



01 Background

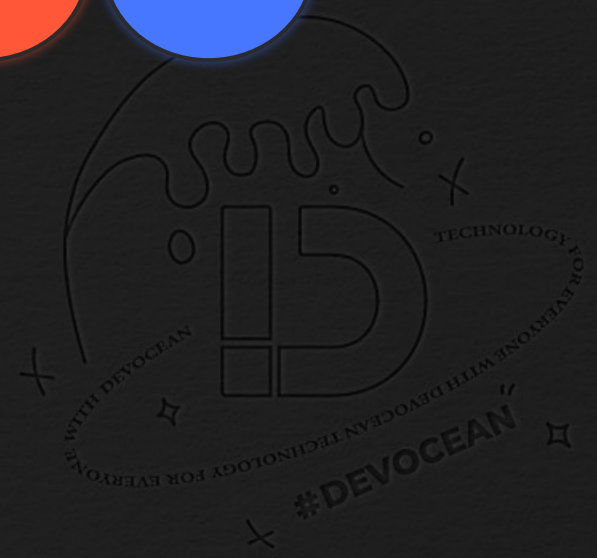
- GPT 출시로 인한 LLM 기술이 관심 상승
- 다양한 OpenLLM 출시
  - Llama 계열, Gemma 계열, Gemini, qwen 계열, Mistral
  - GPT 의 정책으로 인한 OpenLLM 사용 급증
- Private LLM에 대한 관심 증가

02 Problem & Motivation - OpenLLM 모델의 한계(FT 한계)

- OpenLLM 사용에 대한 한계

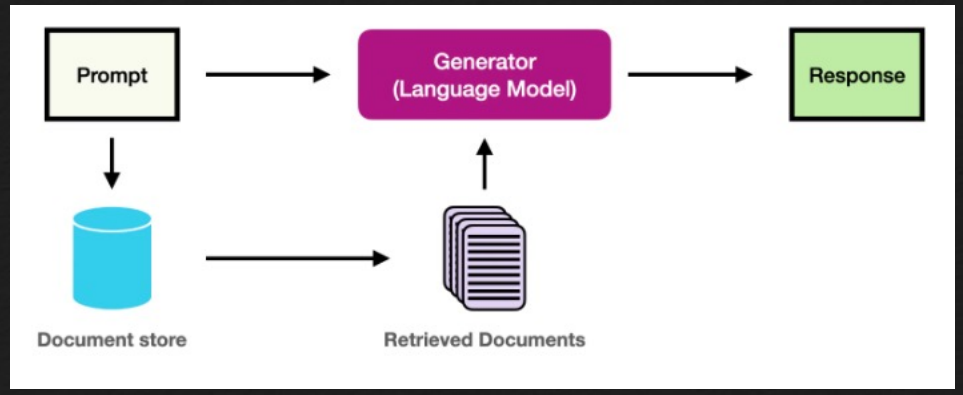
1. **학습 시점 이후의 데이터 없음**
2. **특정 도메인에 대한 데이터 없음**
3. **ML에 대한 지식 및 GPU 필요**

Fine-tuning 필요

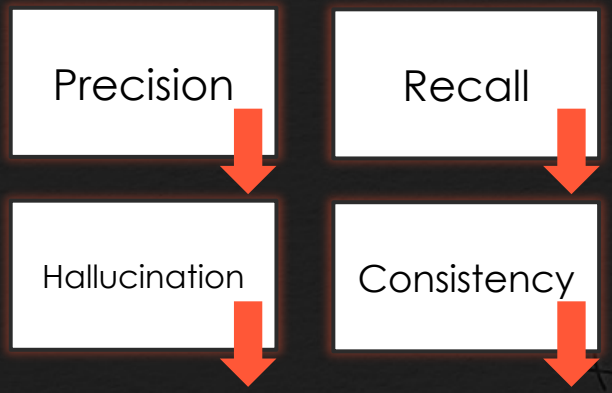


02 Problem & Motivation - RAG 기술과 한계

- RAG 등장과 사용
  - 필요한 지식 보완
  - 적은 리소스
  - 보안이슈 해결
  - LangChain, LlamaIndex 등 편리한 라이브러리 사용
  - 커뮤니티 활성화
  - 사내에 사용하기 적합한 기술

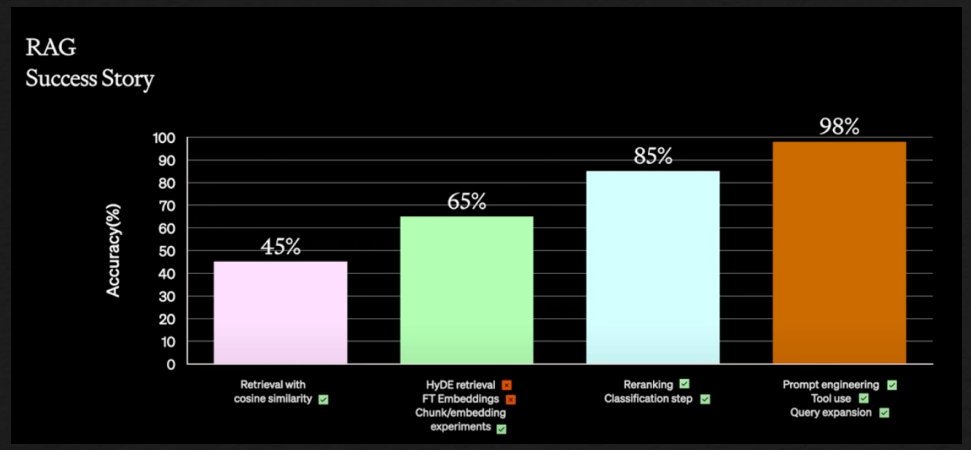


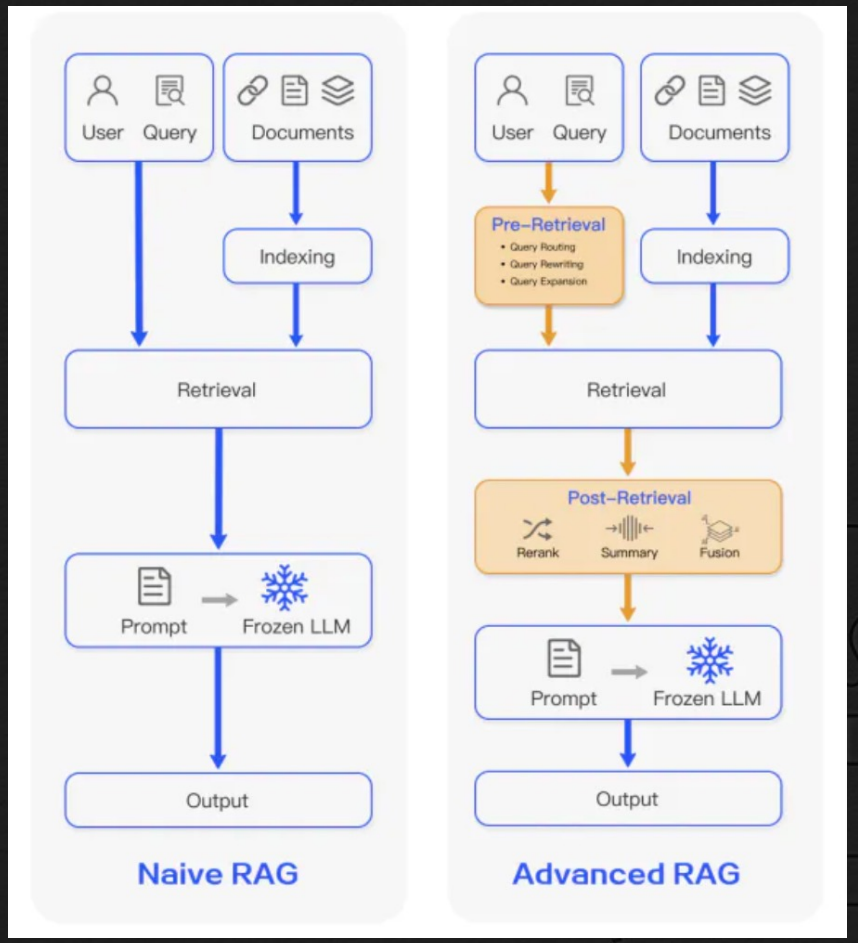
- 하지만 RAG(Naïve RAG) 방식의 한계 존재



03 Solution – Advanced RAG

- What is Advanced RAG?
  - Retrieval 질을 향상 시키기 위한 방법
  - Pre-Retrieval
    - HyDE, MultiQuery, HybridSearch, TextToSQL..
  - Post-Retrieval
    - Rerank

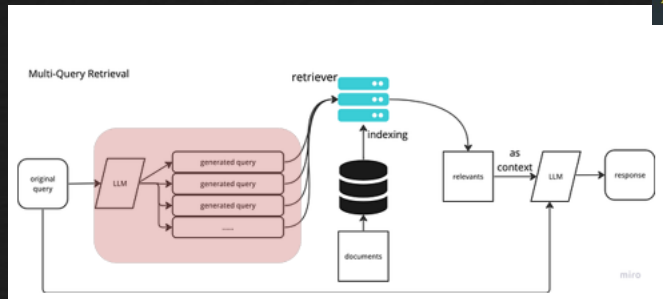


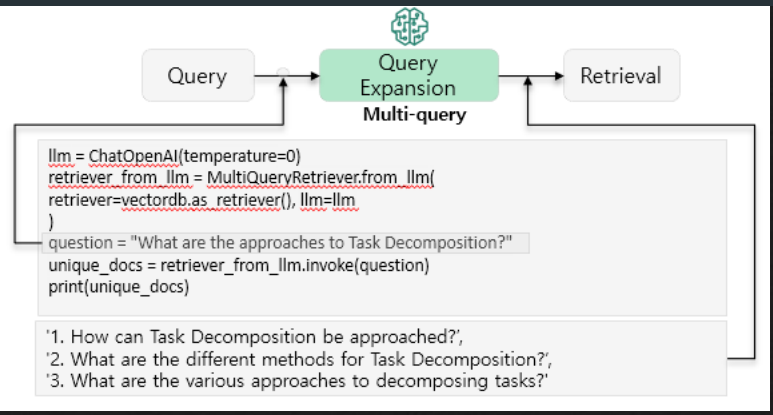


03 Solution – Advanced RAG

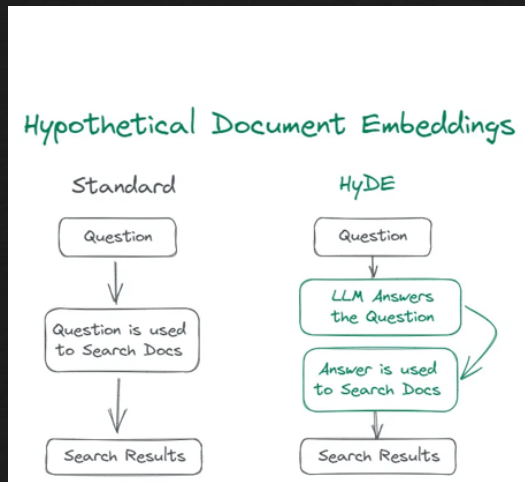
- Pre-Retrieval
  - Multi-Query
    - LLM 을 통해 쿼리를 추가로 생성하여 답을 찾는다.

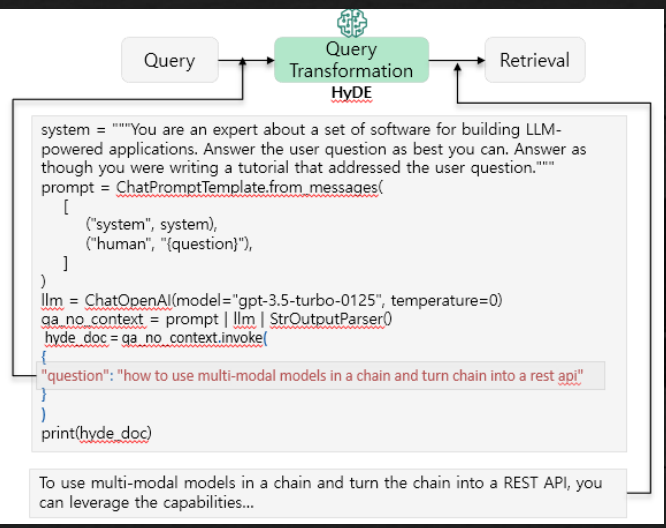
```
prompt_template = PromptTemplate(template="Generate multiple queries for: {question}", input_variables=["question"])
llm_chain = LLMChain(prompt=prompt_template, llm=self.model)

multiquery_retriever = MultiQueryRetriever(
    retriever=faiss_retriever,
    llm_chain=llm_chain,
    num_queries=5  # MultiQueryRetriever에 사용할 쿼리 수 설정
)

self.retrievers = {
    "MultiQuery": multiquery_retriever
}
```





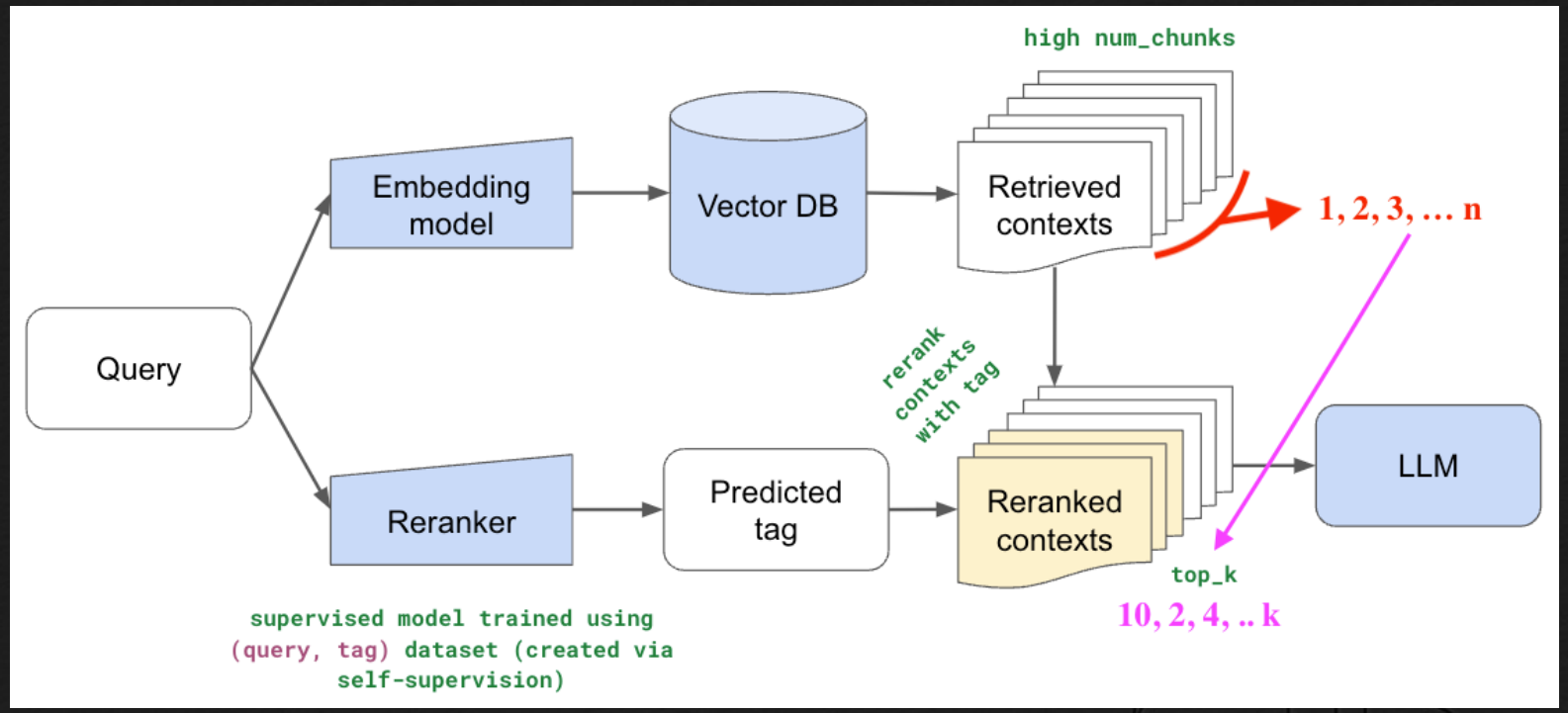
  - HyDE
    - LLM을 통해 가상의 답변을 생성하고 이를 이용





03 Solution – Advanced RAG

- Post-Retrieval
  - Rerank
    - Context를 유사도 계산을 통해 재정렬 하여 전달



03 Solution – RAG 를 만들기 위한 Tool

Framework

haystack by deepset

LlamaIndex

LangChain

Model Serving

OLLAMA

vLLM

Model

Qwen2

Gemma

MISTRAL AI_

Vector DB

chroma

Milvus

Pinecone

OpenSearch

elasticsearch

Evaluation

ragas

AutoRAG

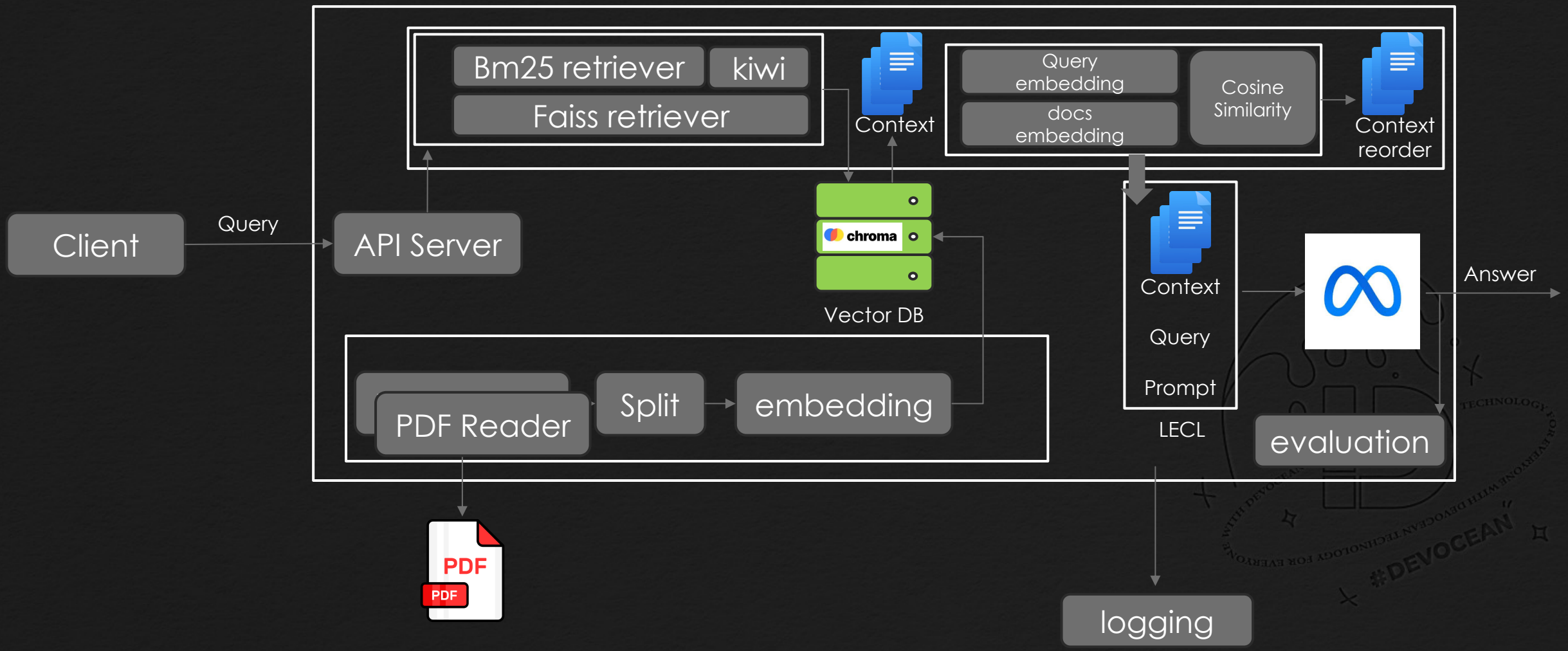
Etc…

Prompt : LangChain Hub

한글 형태소 처리 :
Kiwi , kkma, Okt

03 Solution – Advanced RAG Architecture

- Advanced RAG 를 적용한 구성요소



03 Solution – Advanced RAG 요소

Embedding

- HuggingFace Embedding사용
- 사용 모델 : **BAAI/be-m3**
- CPU 사용
- encode_kwargs 옵션: normalize_embeddings : True
- Nomalize_embedding : True 이점
  - 단순화 된 계산, 유사도 계산 용이
  - 크기 무관성 : 벡터의 의미에 비중
  - 수치 안정성
  - 성능 향상 : 반복 계산에 대한 부하 감소

Code:

```
self.embeddings = HuggingFaceEmbeddings(
    # model_name='jhgan/ko-sroberta-nli',
    model_name='BAAI/bge-m3',
    model_kwargs={'device': 'cpu'},
    encode_kwargs={'normalize_embeddings': True}
)
```

Nomalize_embedding : True

$$\text{Cosine Similarity}(A, B) = \frac{A \cdot B}{\|A\|\|B\|} \rightarrow \text{Cosine Similarity}(A, B) = A \cdot B$$

Kiwi 적용

- LLM 은 영어를 Base로 하는 모델
- 한글어 형태소를 적용하여 한글에 맞게 tokenize 과정 진행

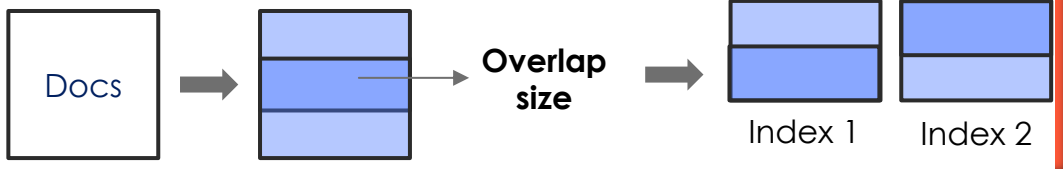
한글어 형태소 처리 적용 전 : '안', '녕' ,'하', '세', '요'
한글어 형태소 처리 적용 후 : '안녕', '하', '세요'

Chunk

- 각 기능 및 문서의 특징에 맞게 사이즈와 **overlap** 변수 지정 필요

Code:

```
self.text_splitter = RecursiveCharacterTextSplitter(
    chunk_size=1024,
    chunk_overlap=100
)
```



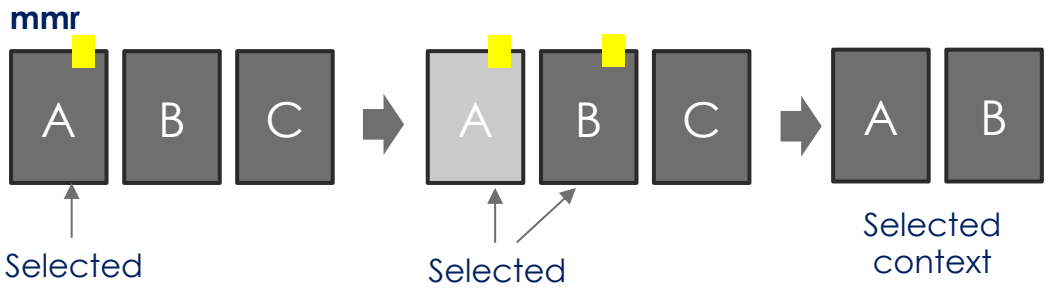
03 Solution – Advanced RAG 요소

Hybrid search(Ensemble Retriever)

- 2개의 리트리버를 조합하여 최적의 결과를 도출
- **(Sparse + Dense) 리트리버 조합 이용**
- LangChain 라이브러리 지원
- 가중치 합이 1.0이 넘지 않게 하여 최적의 구성 찾기
- Search_type : mmr
  - 문서의 관련성과 기존에 선택된 문서와의 균형성을 고려하여 검색결과의 다양성 증진 -> 최적화
  - 이미 선택된 문서는 패널티 적용으로 같은 문서 중복 방지

Code:

```
ensemble_retriever = EnsembleRetriever(
    retrievers=[bm25_retriever, faiss_retriever],
    weights=[0.5, 0.5], # 가중치 설정
    search_type = "mmr"
)
```

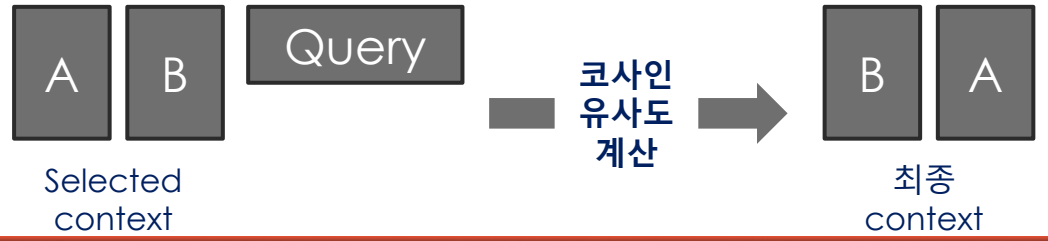


Rerank

- Hybrid search 를 통해 가져온 Context(문서) 재정렬
- Query, Context 를 코사인 유사도 계산을 통해 유사도 기준 재정렬

Code:

```
def rerank(self, query, docs):
    """질문과 문서 리스트를 받아 유사도 기반으로 문서를 재정렬합니다."""
    query_embedding = self.embeddings.embed_query(query)  # 질문 임베딩 생성
    doc_texts = [doc.page_content for doc in docs]  # 문서 텍스트 추출
    doc_embeddings = [self.embeddings.embed_query(doc) for doc in doc_texts]  # 문서 임베딩 생성
    scores = [torch.cosine_similarity(torch.tensor(query_embedding), torch.tensor(doc_embedding),
              for doc_embedding in doc_embeddings]  # 코사인 유사도 계산
    sorted_docs = [doc for _, doc in sorted(zip(scores, docs), key=lambda x: x[0], reverse=True)]
    return sorted_docs
```



03 Solution – Advanced RAG 요소

Prompt

- Prompt은 RAG에서 중요
- 명확하게 표시를 해야 된다.
- 한글로 질문을 해도 명확한 프롬프트가 아니면 영어로 답변
- https://smith.langchain.com/hub 이용

[상황에 따라 추가 된 내용]

- 한글 답변 필요한 경우 : `Answer in Korean.`
- 파이썬 코드 물어본 경우 : `When a user asks a question about Python, please respond with explanations including comments in the code. These comments should help clarify the purpose and functionality of each part of the code to assist the user in understanding.`
- 이전 답변에 대한 추가 질문(chat history) : `When a user includes phrases such as '이전', '이전 답변', '이전 질문' in their query, please refer to the chat history to provide a response. This ensures continuity and relevance in the conversation by acknowledging past interactions.`

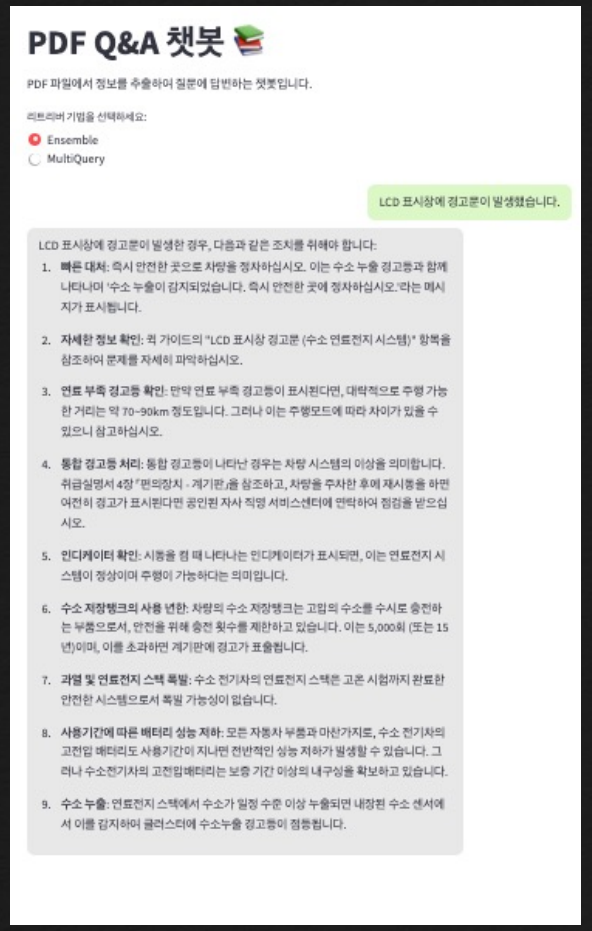
모델

- Llama 3.1 사용
- Temperature 옵션 = 0 사용
  - 답변에 대한 창의성 지수

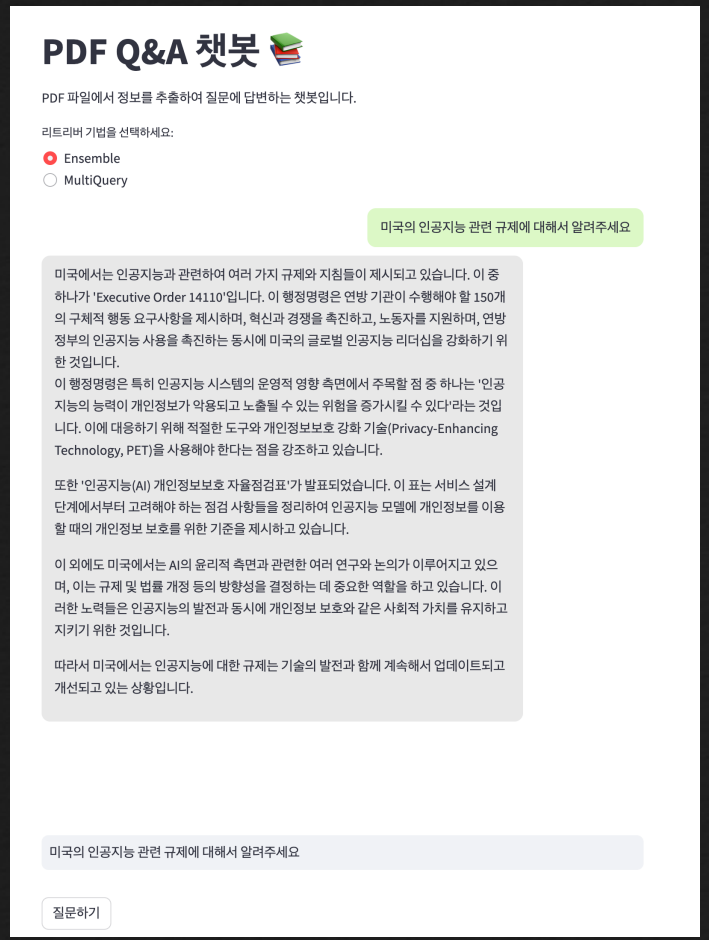
Code:

```
self.model = ChatOllama(model="llama3.1", temperature=0)
```

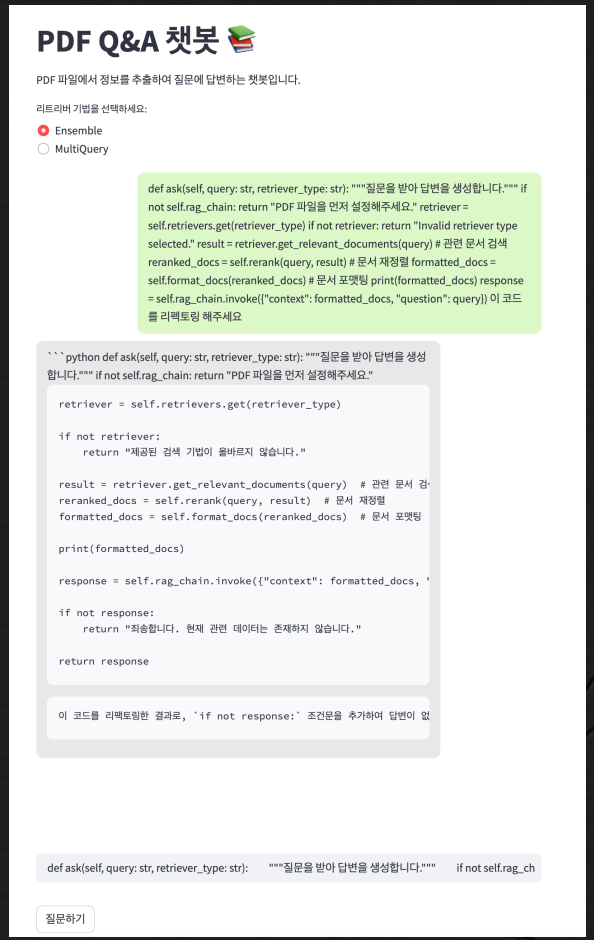
03 Solution - Advanced RAG 결과



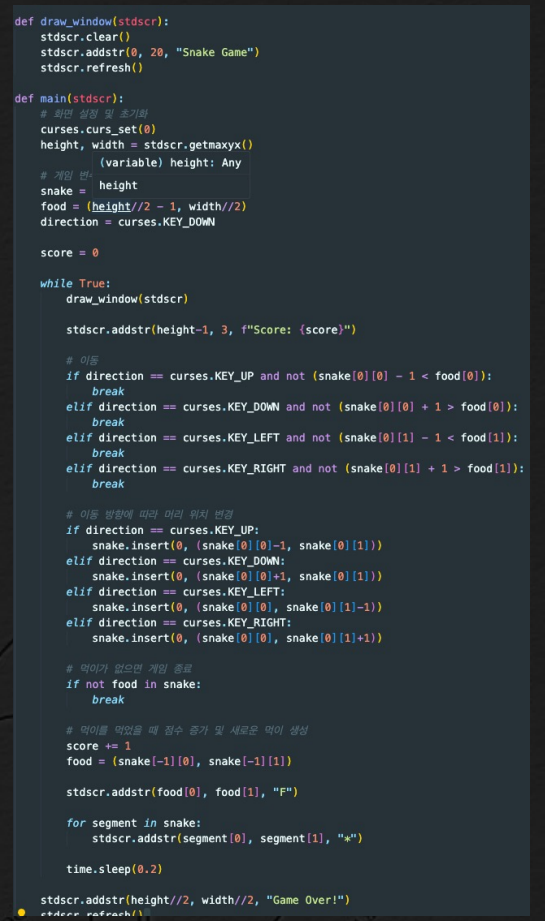
차량 메뉴얼 자료 검색



잡지 자료 검색



코드 리펙토링



코드 생성

03 Solution

```
rag_with_history.invoke(
    # 질문 입력
    {"question": "삼성전자가 만든 생성형 AI 이름은?"},
    # 세션 ID 기준으로 대화를 기록합니다.
    config={"configurable": {"session_id": "rag12345"}},
)
```

✓ 13.3s Python

```
[대화 세션ID]: rag12345

"삼성전자가 만든 생성형 AI 이름은 '삼성 가우스'입니다."
```

이어진 질문 실행

```
rag_with_history.invoke(
    # 질문 입력
    {"question": "이전 답변을 영어로 번역해주세요."},
    # 세션 ID 기준으로 대화를 기록합니다.
    config={"configurable": {"session_id": "rag12345"}},
)
```

✓ 4.5s Python

```
[대화 세션ID]: rag12345

'이전 답변에 대한 영어 번역입니다.\n\n"삼성전자가 만든 생성형 AI 이름은 \'삼성 가우스\'입니다."의 영어 번역은 "The name of the generative AI developed by Samsung
```

챗봇 히스토리

04 Result / Future Work

- Advanced RAG
  - Naïve RAG 가 가지고 있는 문제점 해결 가능
  - 보안적인 이슈 해결
  - LLM을 베이스로 하는 다양한 기능 제작 가능
  - 여러 분야에 응용 가능
    - Q&A, 요약, 코딩 어시스턴트 등..

- Future Work
  - 현재 회사 프로젝트로 진행 -> Hyundai RAG 목표
  - GPU 사용
  - 정형화 된 데이터 준비
  - 다양한 기능을 하는 RAG 기반 엔진 도입
  - RAFT를 통한 성능 개선

05 스터디를 마치며

- 좋은 사람들과의 만남
- 기술적인 발전
- 동기부여
- 앞으로 나의 방향성에 대해 알 수 있는 계기