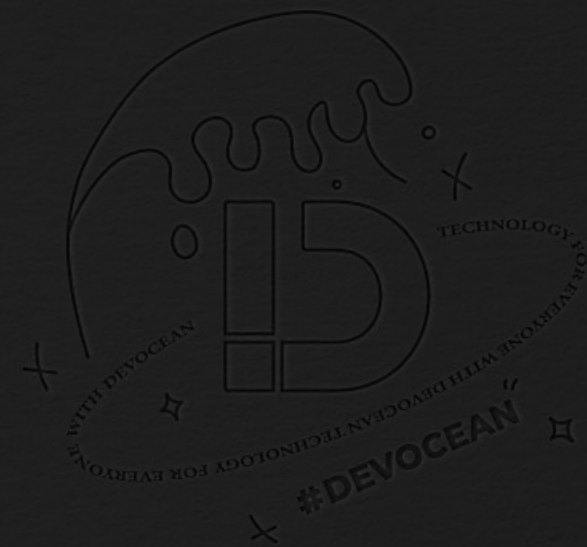




OpenLLM / RAG 업무 필요 사례

현대오토에버 이길호



목차

01 Background

02 Problem & motivation

03 Solution

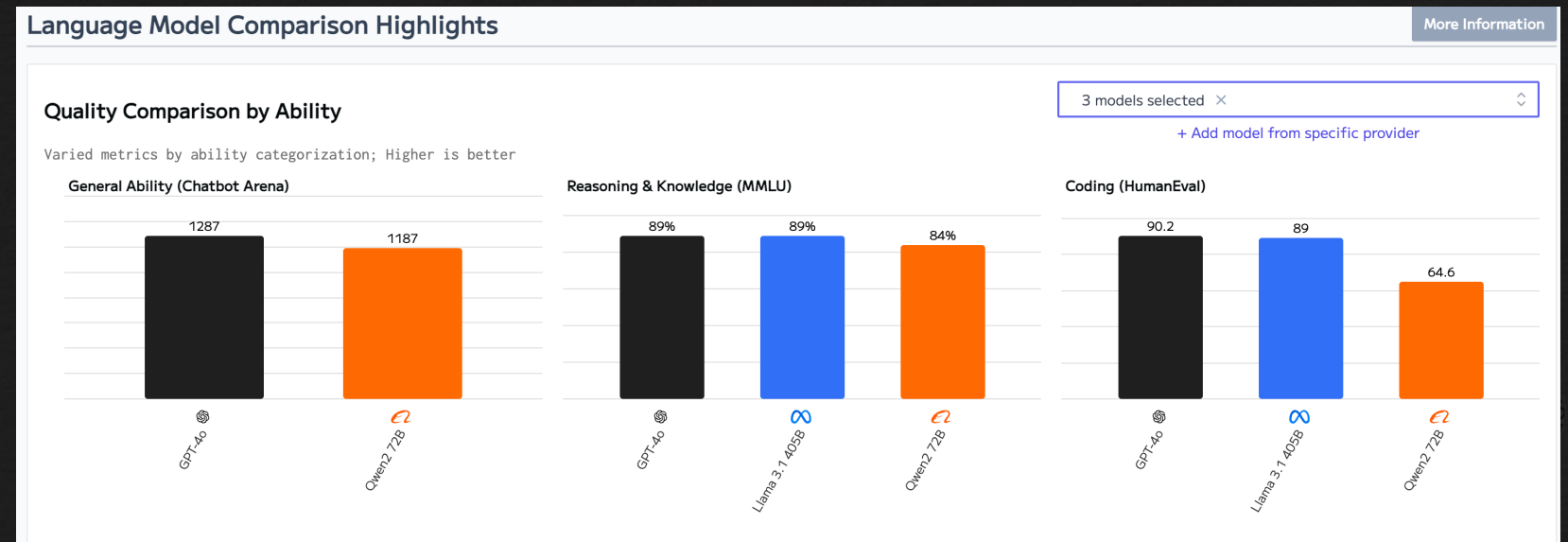
04 Result / Future Work

05 스터디를 마치며..



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

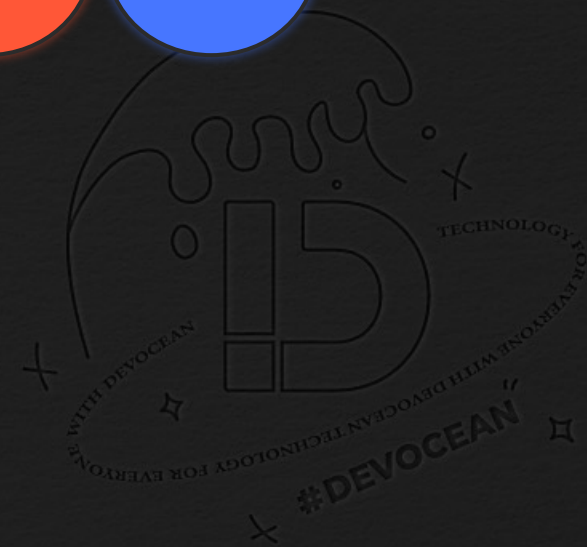
3 ML에 대한 지식 및 GPU 필요

Fine-tuning
필요

시간

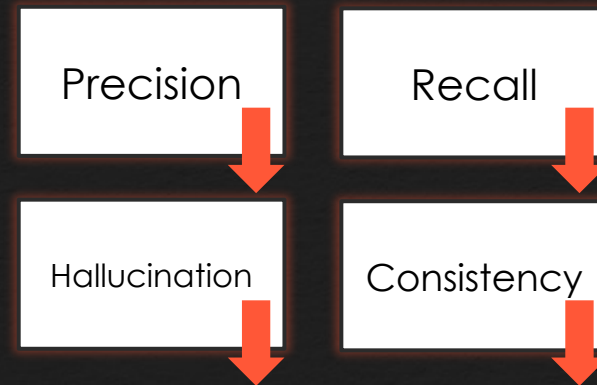
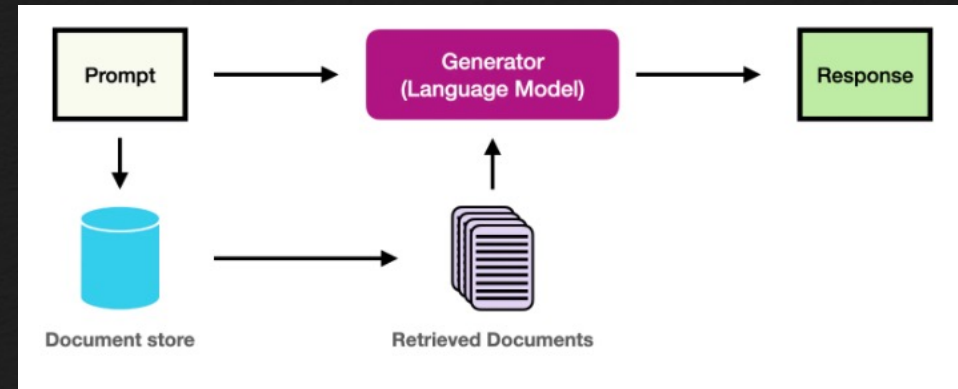
비용

자원



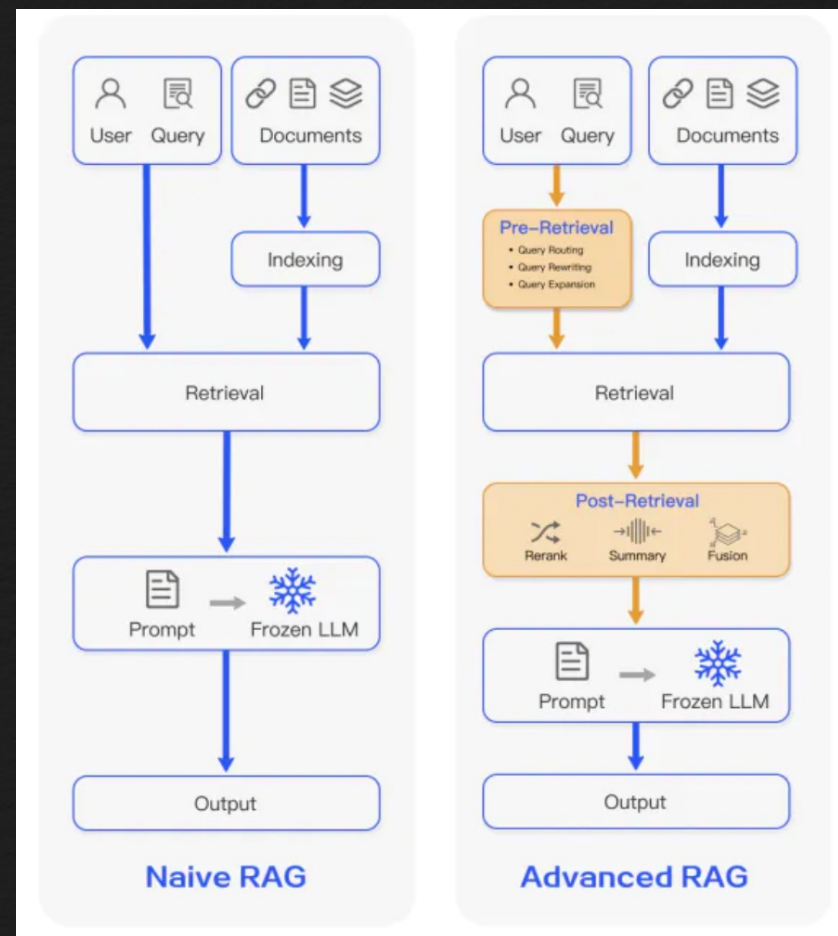
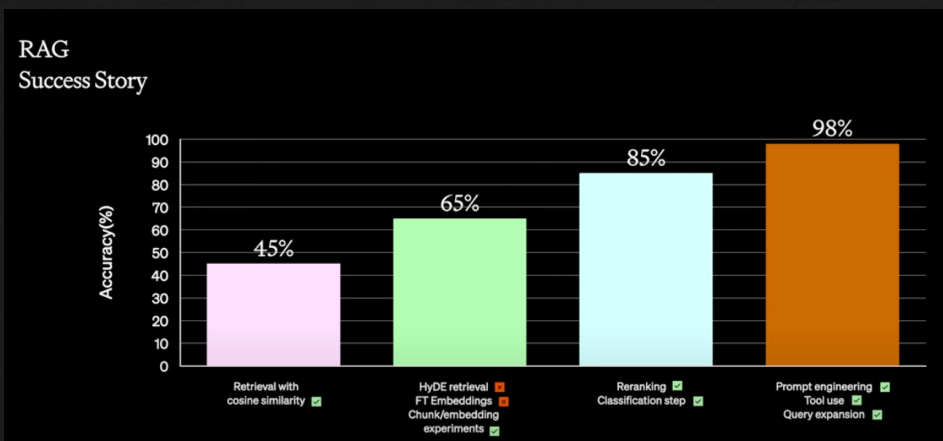
02 Problem & Motivation - RAG 기술과 한계

- 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

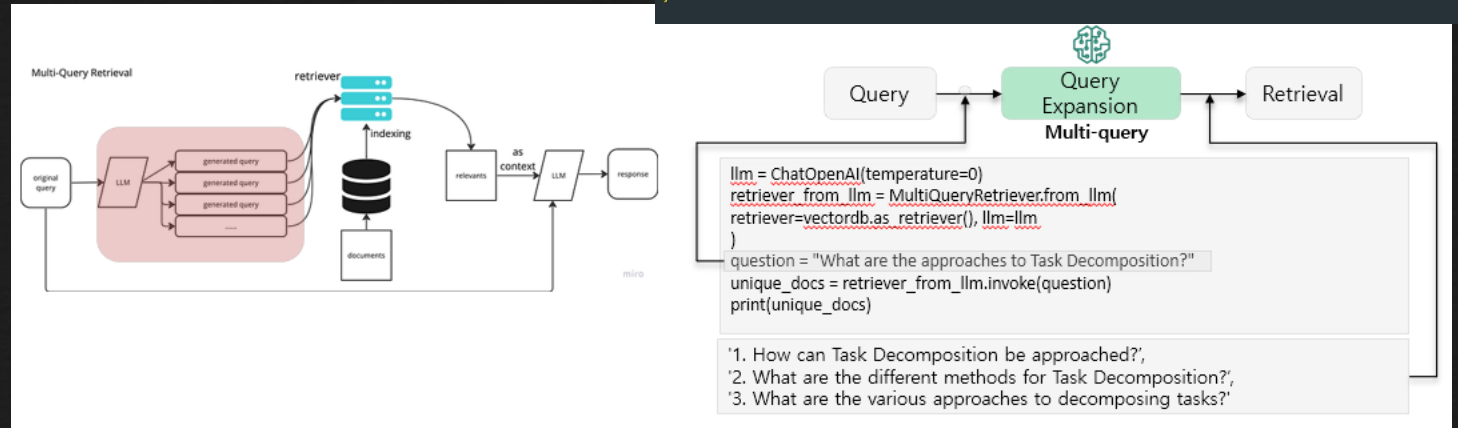


03 Solution – Advanced RAG

- Pre-Retrieval

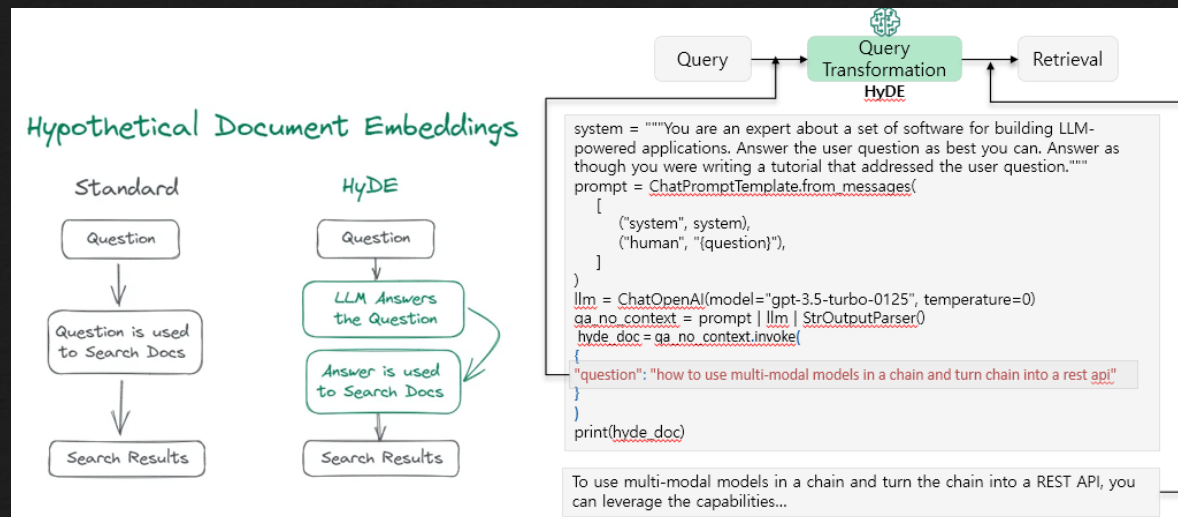
- Multi-Query

- LLM 을 통해 쿼리를 추가로 생성하여 답을 찾는다.



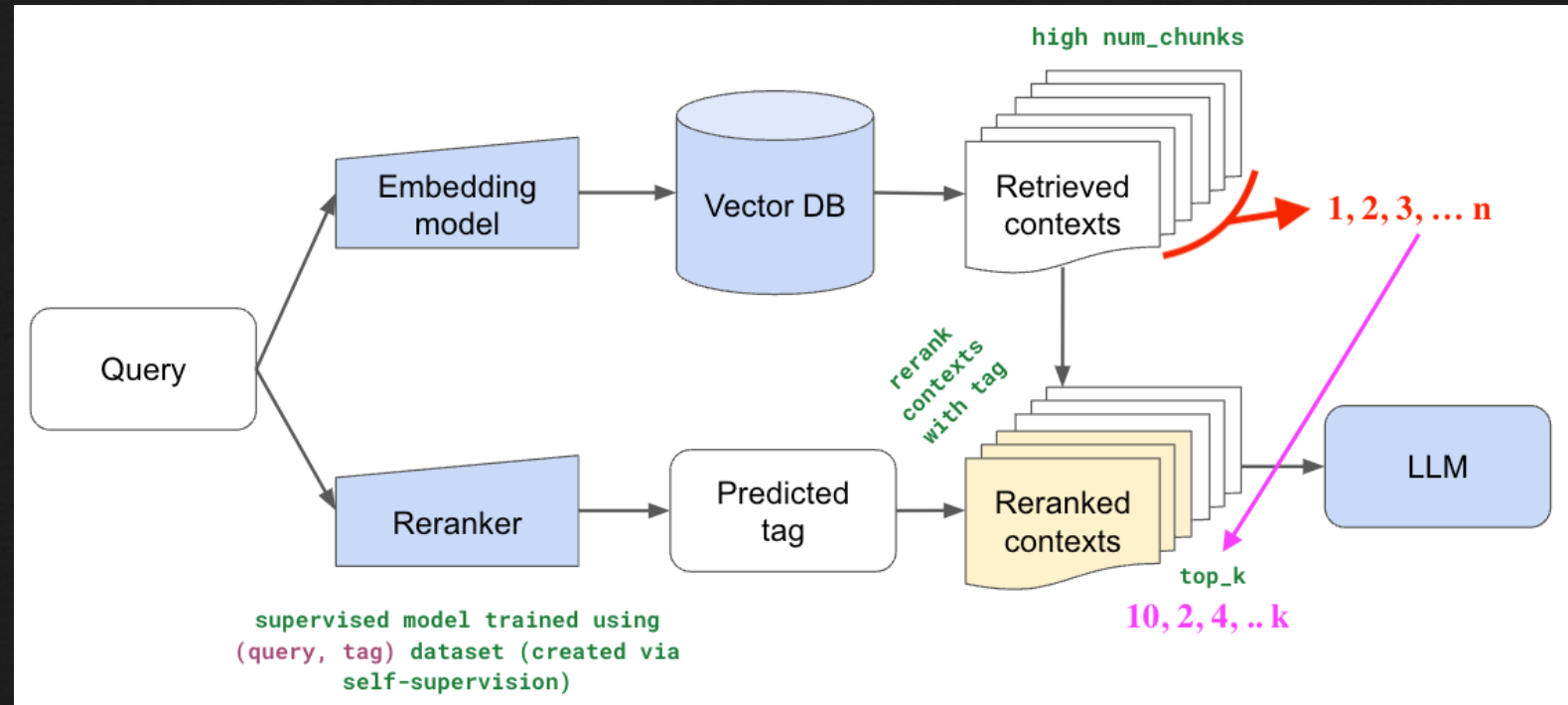
- HyDE

- 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



LlamaIndex



Model Serving



Model



Qwen2



Vector DB



elasticsearch

Evaluation



ragas

AutoRAG

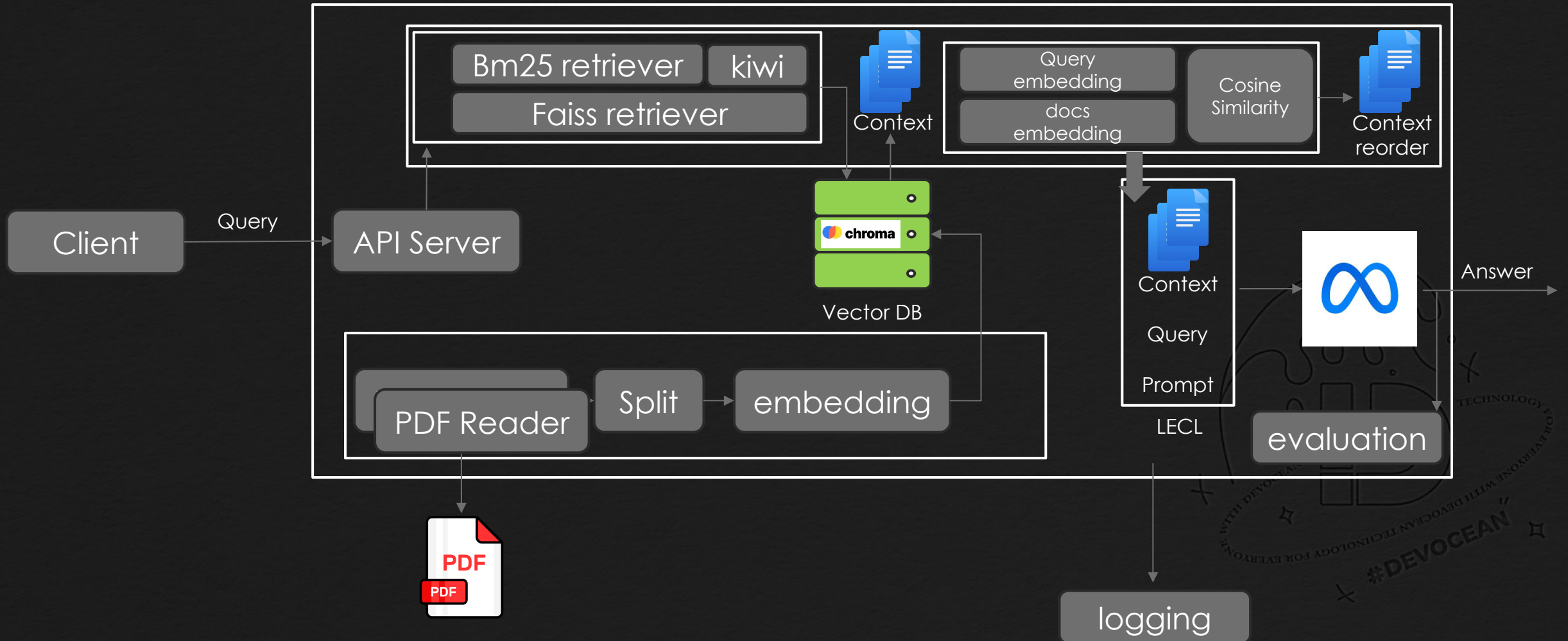
Etc...

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 Embedding 사용
- 사용 모델 : **BAAI/bge-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

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

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

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

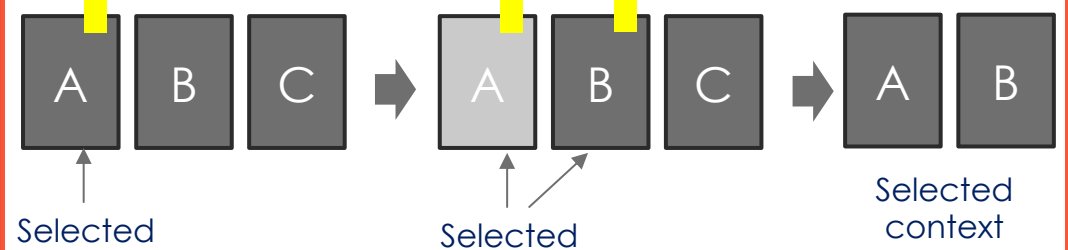
Hybrid search(Ensemble Retriever)

- 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"  
)
```

mmr

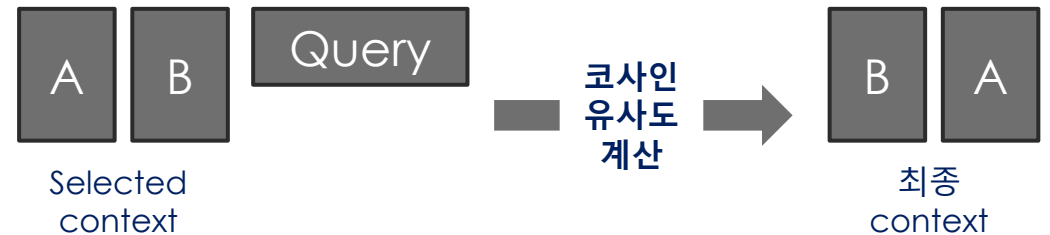


Rerank

- 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

PDF Q&A 챗봇

PDF 파일에서 정보를 추출하여 질문에 답변하는 챗봇입니다.

리트리버 기법을 선택하세요:

- ☒ Ensemble
- ☐ MultiQuery

LCD 표시상에 경고문이 발생했습니다.

LCD 표시상에 경고문이 발생한 경우, 다음과 같은 조치를 취해야 합니다:

- 빠른 대처:** 즉시 안전한 곳으로 차량을 정차하십시오. 이는 수소 누출 경고등과 함께 나타나며 '수소 누출이 감지되었습니다. 즉시 안전한 곳에 정차하십시오.'라는 메시지가 표시됩니다.
- 자세한 정보 확인:** 각 가이드의 "LCD 표시상 경고문 (수소 연료전지 시스템)" 항목을 참조하여 문제를 자세히 파악하십시오.
- 연료 부족 경고등 확인:** 만약 연료 부족 경고등이 표시된다면, 대략적으로 주행 가능한 거리는 약 70~90km 정도입니다. 그러나 이는 주행모드에 따라 차이가 있을 수 있으니 참고하십시오.
- 통합 경고등 처리:** 통합 경고등이 나타난 경우는 차량 시스템의 이상을 의미합니다. 취급설명서 4장 "편의장치 - 계기판"을 참조하고, 차량을 주차한 후에 재시동을 하면 여전히 경고가 표시된다면 공인된 자사 직영 서비스센터에 연락하여 점검을 받으십시오.
- 인디케이터 확인:** 시동을 켰 때 나타나는 인디케이터가 표시되면, 이는 연료전지 시스템이 정상이며 주행을 가능하다는 의미입니다.
- 수소 저장탱크의 사용 년한:** 차량의 수소 저장탱크는 고압의 수소를 수시로 충전하는 부분으로서, 안전을 위해 충전 횟수를 제한하고 있습니다. 이는 5,000회 (또는 15년)이며, 이를 초과하면 계기판에 경고가 표출됩니다.
- 과열 및 연료전지 스택 폭발:** 수소 전기자의 연료전지 스택은 고온 시험까지 완료한 안전한 시스템으로서 폭발 가능성이 없습니다.
- 사용기간에 따른 배터리 성능 저하:** 모든 자동차 부품과 마찬가지로, 수소 전기차의 고전압 배터리도 사용기간이 지나면 전반적인 성능 저하가 발생할 수 있습니다. 그러나 수소전기차의 고전압배터리는 보통 기간 이상의 내구성을 확보하고 있습니다.
- 수소 누출:** 연료전지 스택에서 수소가 일정 수준 이상 누출되면 내장된 수소 센서에 서 이를 감지하여 클러스터에 수소누출 경고등이 점등됩니다.

PDF Q&A 챗봇

PDF 파일에서 정보를 추출하여 질문에 답변하는 챗봇입니다.

리트리버 기법을 선택하세요:

- ☒ Ensemble
- ☐ MultiQuery

미국의 인공지능 관련 규제에 대해서 알려주세요

미국에서는 인공지능과 관련하여 여러 가지 규제와 지침들이 제시되고 있습니다. 이 중 하나가 'Executive Order 14110'입니다. 이 행정명령은 연방 기관이 수행해야 할 150개의 구체적 행동 요구사항을 제시하며, 혁신과 경쟁을 촉진하고, 노동자를 지원하며, 연방 정부의 인공지능 사용을 촉진하는 동시에 미국의 글로벌 인공지능 리더십을 강화하기 위한 것입니다. 이 행정명령은 특히 인공지능 시스템의 운영적 영향 측면에서 주목할 점 중 하나는 '인공지능의 능력이 개인정보가 악용되고 노출될 수 있는 위험을 증가시킬 수 있다'라는 것입니다. 이에 대응하기 위해 적절한 도구와 개인정보보호 강화 기술(Privacy-Enhancing Technology, PET)을 사용해야 한다는 점을 강조하고 있습니다.

또한 '인공지능(AI) 개인정보보호 자율점검표'가 발표되었습니다. 이 표는 서비스 설계 단계에서부터 고려해야 하는 점검 사항들을 정리하여 인공지능 모델에 개인정보를 이용할 때의 개인정보 보호를 위한 기준을 제시하고 있습니다.

이 외에도 미국에서는 AI의 윤리적 측면과 관련한 여러 연구와 논의가 이루어지고 있으며, 이는 규제 및 법률 개정 등의 방향성을 결정하는 데 중요한 역할을 하고 있습니다. 이러한 노력들은 인공지능의 발전과 동시에 개인정보 보호와 같은 사회적 가치를 유지하고 지키기 위한 것입니다.

따라서 미국에서는 인공지능에 대한 규제는 기술의 발전과 함께 계속해서 업데이트되고 개선되고 있는 상황입니다.

미국의 인공지능 관련 규제에 대해서 알려주세요

질문하기

PDF Q&A 챗봇

PDF 파일에서 정보를 추출하여 질문에 답변하는 챗봇입니다.

리트리버 기법을 선택하세요:

- ☒ Ensemble
- ☐ MultiQuery

```
def ask(self, query: str, retriever_type: str): """질문을 받아 답변을 생성합니다.""" if not self.rag_chain: return "PDF 파일을 먼저 설정해주세요." retriever = self.retrievers.get(retriever_type) if not retriever: return "Invalid retriever type selected." result = retriever.get_relevant_documents(query) # 관련 문서 검색 reranked_docs = self.rerank(query, result) # 문서 재정렬 formatted_docs = self.format_docs(reranked_docs) # 문서 포맷팅 print(formatted_docs, "question": query)) 이 코드를 리팩토링 해주세요
```

```
python def ask(self, query: str, retriever_type: str): """질문을 받아 답변을 생성합니다.""" if not self.rag_chain: return "PDF 파일을 먼저 설정해주세요."
```

```
retriever = self.retrievers.get(retriever_type)

if not retriever:
    return "재levant 검색 기법이 올바르지 않습니다."
```

```
result = retriever.get_relevant_documents(query) # 관련 문서 검색
reranked_docs = self.rerank(query, result) # 문서 재정렬
formatted_docs = self.format_docs(reranked_docs) # 문서 포맷팅

print(formatted_docs)

response = self.rag_chain.invoke({"context": formatted_docs, "question": query})

if not response:
    return "죄송합니다. 현재 관련 데이터는 존재하지 않습니다."
```

```
return response
```

이 코드를 리팩토링한 결과로, 'if not response:' 조건문을 추가하여 답변이 없

```
def ask(self, query: str, retriever_type: str): """질문을 받아 답변을 생성합니다.""" if not self.rag_chain:
```

질문하기

```
def draw_window(stdscr):
    stdscr.clear()
    stdscr.addstr(0, 20, "Snake Game")
    stdscr.refresh()

def main(stdscr):
    # 화면 설정 및 초기화
    curses.curs_set(0)
    height, width = stdscr.getmaxyx()
    (variable) height: Any
    # 게임 변수
    snake = (height//2, width//2)
    food = (height//2 - 1, width//2)
    direction = curses.KEY_DOWN

    score = 0

    while True:
        draw_window(stdscr)

        stdscr.addstr(height-1, 3, f"Score: {score}")

        # 이동
        if direction == curses.KEY_UP and not (snake[0][0] - 1 < food[0]):
            break
        elif direction == curses.KEY_DOWN and not (snake[0][0] + 1 > food[0]):
            break
        elif direction == curses.KEY_LEFT and not (snake[0][1] - 1 < food[1]):
            break
        elif direction == curses.KEY_RIGHT and not (snake[0][1] + 1 > food[1]):
            break

        # 이동 방향에 따라 머리 위치 변경
        if direction == curses.KEY_UP:
            snake.insert(0, (snake[0][0]-1, snake[0][1]))
        elif direction == curses.KEY_DOWN:
            snake.insert(0, (snake[0][0]+1, snake[0][1]))
        elif direction == curses.KEY_LEFT:
            snake.insert(0, (snake[0][0], snake[0][1]-1))
        elif direction == curses.KEY_RIGHT:
            snake.insert(0, (snake[0][0], snake[0][1]+1))

        # 먹이가 없으면 게임 종료
        if not food in snake:
            break

        # 먹이를 먹었을 때 점수 증가 및 새로운 먹이 생성
        score += 1
        food = (snake[-1][0], snake[-1][1])

        stdscr.addstr(food[0], food[1], "F")

        for segment in snake:
            stdscr.addstr(segment[0], segment[1], "x")

        time.sleep(0.2)

    stdscr.addstr(height//2, width//2, "Game Over!")
    stdscr.refresh()
```

차량 메뉴얼 자료
검색

잡지 자료 검색

코드 리팩토링

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

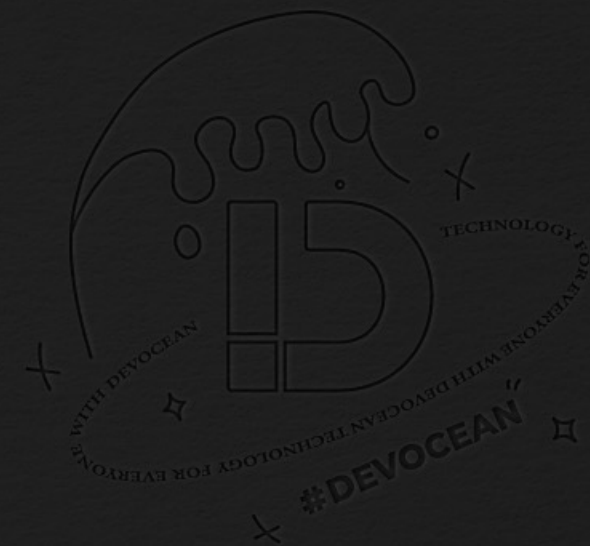
```
rag_with_history.invoke(  
    # 질문 입력  
    {"question": "이전 답변을 영어로 번역해주세요."},  
    # 세션 ID 기준으로 대화를 기록합니다.  
    config={"configurable": {"session_id": "rag12345"}},  
)
```

✓ 4.5s Python

[대화 세션ID]: rag12345

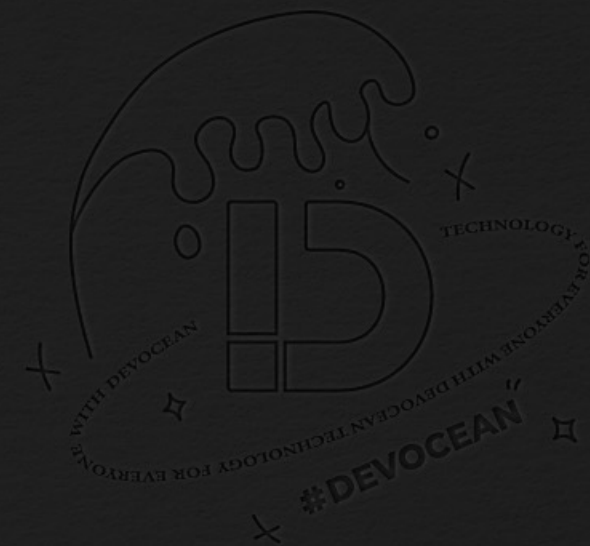
'이전 답변에 대한 영어 번역입니다.\n\n"삼성전자가 만든 생성형 AI 이름은 \'삼성 가우스\'입니다."'의 영어 번역은 "The name of the generative AI developed by Samsung is \'Samsung Gaus\'."

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

