Wrap-up Report

1. 프로젝트 개요

1.1 프로젝트 소개

기획배경

일반 투자자들은 대부분 증권사의 투자의견과 목표주가만을 참고합니다. **하지만 주식 시장은 이보다 훨씬 복잡한 시스템**입니다. 기업의 내재가치, 시장심리, 거시경제 상황, 산업동향 등 수많은 변수들이 **서로 영향을 주고받으며 주가를 형성**하기 때문입니다.
특히,
개인 투자자의 경우에 이러한 다양한 변수들을 실시간으로 모니터링하고 **투자 결정을 내리기가 매우 어렵습니다.** 거시경제 지표를 보더라도 환율, 금리, 원자재 가격 등의 상호 연관성을 파악하기가 쉽지 않습니다. 기업의 재무제표나 기술적 분석 또한 전문적인 지식이 필요합니다.
이러한 문제를 해결하기 위해서
"거시경제/미시경제/차트/뉴스 등 각 분야에 특화된 AI 에이전트를 만들고 복잡한 의사결정을 대신해줄 수는 없을까?" 라는 아이디어를 가지고 이 프로젝트를 시작하게 되었습니다.

프로젝트 주제 및 주요 서비스

저희는 투자하고 싶은 기업의 중요 정보를 쉽고 빠르게 **하루 3번,**
진행된 매매 포지션과 함께
핵심 정보가 포함된 투자 리포트를 카톡알람 해주는 서비스를 제공합니다.

서비스의 3가지 핵심 기능

✅ AI 멀티 에이전트 기반의 **주식 매매관리 서비스**

✅ 각 분야에서 전문화 된 AI 에이전트가 특화된 전략으로 **기업의 주식 현황을 분석하고 투자자에게 매매 포지션을 제공**합니다.

✅ RAG 시스템으로 출력되는 증권사 보고서의 정보와 웹 데이터(재무재표/거시경제/미시경제/차트 등)를 활용하여 **핵심 리포트를 사용자에게 제공**합니다.

1.2 팀 소개 및 역할

김현서 : GraphRAG 실험, LLM api 실험, 모델 프롬프트 작성, logit 기반 평가 로직 설계

이재룡 : DB 설계, Reranking, Model & Prompt, 매매관리 Agent 설계

이정인 : 서비스 기획, PDF 파싱 및 청킹, 임베딩 테스트, Agent 구현, 호가창 API 설계

이현풍 : PDF 파싱 및 청킹, Reranking, Agent 구현, LangGraph 스켈레톤 코드 작성, Multi-Agent 기획 및 설계, 발표 기획 및 제작

임한택 : 서비스 기획, PDF 파싱 및 청킹 고도화, RAG 파이프라인 고도화, G-EVAL Test, Model & Prompt 설계 및 실험, Reranker 실험

최현우 : PM, 서비스 기획, DB 설계, 임베딩 기획 및 테스트, RAG 파이프라인 기획 및 고도화, 서비스 아키텍처 기획 및 구현, 프론트엔드, 백엔드 기획 및 구현

1.3 개발/협업 환경

하드웨어

Tesla V100 32GB * 4EA

AWS Lightsail VM 2EA, Database 1EA

소프트웨어 및 주요 라이브러리

Github : 팀 내의 커밋 컨벤션 규칙을 작성하여 통일된 모습으로 쉽게 변경 사항을 찾을 수 있도록 했습니다. 기본 템플릿을 dev 브랜치로 둔 뒤 experiement 브랜치에서 개발 및 실험을 진행하였습니다. 새로운 기능을 개발하거나 실험할 때는 분기를 만들어서 병합을 진행하였습니다.

Notion : 메인 TASK를 보드에 두고 업무를 할당했으며 준비/진행중/완료로 나누어서 개발 상황을 가시적으로 공유했습니다. 그리고 프로젝트 수행에 필요한 정리된 문서를 통해 정보와 자료의 공유 공간으로 사용했습니다. 줌 및 슬랙 회의록을 작성해서 개인별 진행 상황과 레퍼런스 등을 공유했습니다.

Langsmith : LangGraph를 활용해서 Langchain으로 모듈화 된 Agent를 구축하는 프로젝트로서 리트리버가 어떤 문서를 불러왔는지, 모델이 어떤 답변을 했는지 등을 팀원들끼리 공유하기 위해서 사용했습니다.

```
langchain = "0.3.15"
langgraph = "0.2.19"
sentence-transformers = "^3.3.1"
beautifulsoup4 = "4.12.3"
yfinance = "^0.2.52"
pandas = "^2.2.3"
streamlit = "^1.17.0"
gradio = "5.12.0"
mojito2 = "^0.1.6" (한국투자증권 API)
elasticsearch = "^8.17.1"
fastapi = "^0.115.8"
```

2. 프로젝트 수행 절차 및 방법

2.1 프로젝트 로드맵

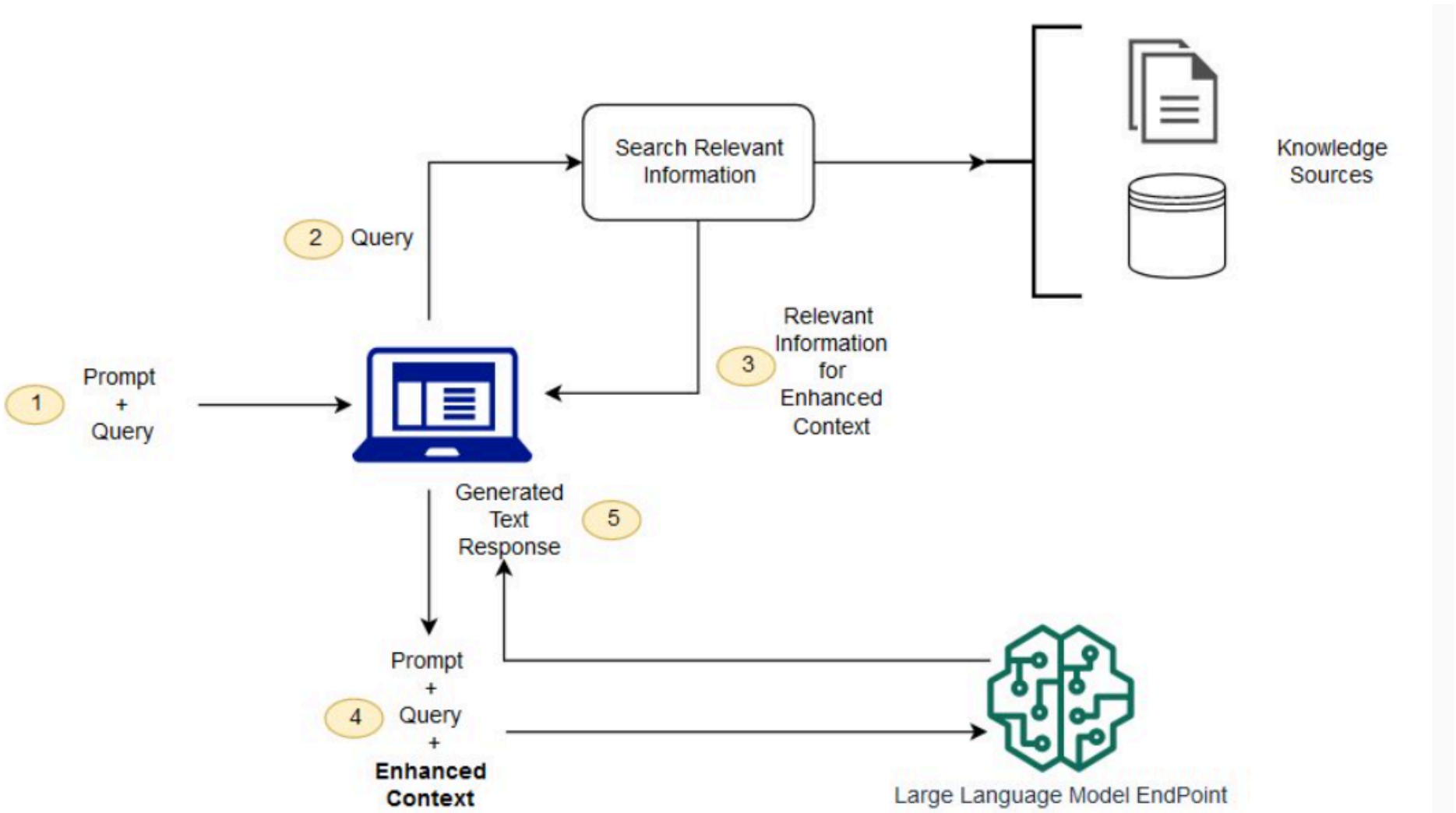
저희 팀은 G-EVAL 정량 평가를 위해서 100개의 PDF 증권사 리포트를 토대로 RAG 파이프라인 구축부터 시작했습니다. 이후 RAG 시스템을 하나의 에이전트로 구축하고 특정 기업에 대해 전문가의 의견을 불러올 수 있도록 Multi-Agent에 병합시켰습니다. 작업 순서는 다음과 같습니다.

1. 기업 분석 Agent를 위한 RAG 파이프라인 설계
2. RAG TASK 별 개발 진행(Parsing → Chunking → Embedding → Vector → Retriever → Reranker → Model & Prompt)
3. Multi-Agent 구축
4. LangGraph로 워크플로우 형성
5. DB 구축
6. UI/UX 및 세부 기능 구현
7. 배포

2.2 세부 내용

RAG 시스템 개발

LLM은 잘못된 정보를 출력하거나 최신 정보가 결여되어 있다는 점에서 한계를 가집니다. 하지만 검색이 보강된 생성 방식(Retrieval Augmented Generation, **RAG)으로 LLM의 단점을 보완**할 수 있습니다. 저희 팀은 증권사 리포트 PDF 파일의 내용을 잘 파싱해서 모델이 증권사 전문 지식이 보강된 답변을 할 수 있도록 만드는 것이 RAG 시스템 개발의 목표였습니다. 이 과정에서 정량적인 비교를 위해 G-EVAL 테스트 데이터셋을 구축하고 Retrieval 평가(20점 만점)와 Generation 평가(30점 만점)하여 성능을 비교했습니다.



전통적인 RAG 파이프라인
출처:
https://aws.amazon.com/ko/what-is/retrieval-augmented-generation/

- **문서 전처리 (파싱 및 청킹) :** PDF의 내용을 모델에 입력하기 위해서 구조화된 텍스트로 파싱하는 작업이 필요합니다. 이때 저희 팀이 가장 중요하게 생각한 것은 테이블, 차트 데이터의 맥락적 정보 유지입니다. 테이블과 차트 데이터는 그 자체만으로 정보를 갖기는 어렵습니다. 전후로 있는 caption, header, paragraph 정보들과 상호 연관되어 있습니다. 따라서 맥락적인 정보를 유지시키기 위해 4단계로 나누어 파싱 및 청킹을 진행했습니다.

  1. 1차 파싱 요소 활용 : 먼저 Upstage Document Parser를 사용해서 100개의 PDF 파일을 파싱했습니다. 파싱한 결과는 PDF의 layout 별로 ID가 부여되고 카테고리(table, chart, figure, paragraph, header, caption 등), base_64_encdoing 등 다양한 메타데이터를 포함합니다. 이 경우 행과 열만 존재하는 테이블이 파싱(아래의 이미지중 검정색 부분)되고 그외 단위 정보나 캡션 정보(아래의 이미지중 파란색 부분)는 따로 저장이 됩니다. 그래서 테이블 정보를 유지하기 위해서 header, caption, heading1과 같은 요소들을 테이블 및 차트에 병합시키는 작업을 진행했습니다.

  2. 2차 파싱 Chunk : PDF 내에 나와있는 문단 텍스트(paragraph)에 대해서만 chunk_size=600, chunk_overlap=50으로 청킹을 진행합니다.

CJ제일제당 단위: 십억원

결산	2022A	2023A	2024F	2025F	2026F
영업활동 현금흐름	1,627	2,445	2,652	2,849	3,072
당기순이익	803	559	705	740	919
비현금항목의 가감	2,328	2,273	2,783	2,810	2,894
감가상각비	1,254	1,380	1,383	1,410	1,456
외환손익	0	0	28	28	28
지분법평가손익	0	0	-19	-19	-19
기타	1,074	893	1,391	1,391	1,429
자산부채의 증감	-1,134	-155	-159	-23	-23
기타현금흐름	-369	-233	-678	-679	-717

3. 3차 파싱 메타데이터 추가 : 메타데이터를 임베딩 될 page_content에 추가해서 맥락 정보를 추가합니다. 예를 들어서 메타데이터에 증권사 이름, 발행날짜, 회사 이름이 있다면, 이를 '{발행날짜}에 {증권사 이름}에서 발행한 {회사 이름}에 관한 주식 리포트'로 다음과 같이 추가합니다. {메타데이터 정보} + {원래 본문 내용}

page_content = 20241113에 교보증권에서 발행한 CJ제일제당에 관한 레포트에서 나온 내용. (3차 파싱 메타데이터 활용)

CJ제일제당 단위: 십억원

결산	2022A	2023A	2024F	2025F	2026F
영업활동 현금흐름	1,627	2,445	2,652	2,849	3,072
당기순이익	803	559	705	740	919
비현금항목의 가감	2,328	2,273	2,783	2,810	2,894
감가상각비	1,254	1,380	1,383	1,410	1,456
외환손익	0	0	28	28	28
지분법평가손익	0	0	-19	-19	-19
기타	1,074	893	1,391	1,391	1,429
자산부채의 증감	-1,134	-155	-159	-23	-23
기타현금흐름	-369	-233	-678	-679	-717

4. 4차 파싱 요약본 추가 : 각 테이블, 차트에 대한 요약을 LLM으로 생성해서 임베딩 될 page_content에 추가합니다.

page_content = 20241113에 교보증권에서 발행한 CJ제일제당에 관한 레포트에서 나온 내용.

summary : 2022년 영업활동 현금흐름 1627 삽억원... (4차 요약본 추가)

CJ제일제당 단위: 십억원

결산	2022A	2023A	2024F	2025F	2026F
영업활동 현금흐름	1,627	2,445	2,652	2,849	3,072
당기순이익	803	559	705	740	919
비현금항목의 가감	2,328	2,273	2,783	2,810	2,894
감가상각비	1,254	1,380	1,383	1,410	1,456
외환손익	0	0	28	28	28
지분법평가손익	0	0	-19	-19	-19
기타	1,074	893	1,391	1,391	1,429
자산부채의 증감	-1,134	-155	-159	-23	-23
기타현금흐름	-369	-233	-678	-679	-717

- **임베딩 및 DB 구축 :** 임베딩(Embedding)은 텍스트 데이터를 숫자로 이루어진 벡터로 변환하는 과정입니다. 이를 통해 텍스트 데이터를 벡터 공간에서 다룰 수 있게 되며 텍스트 간의 유사성을 계산하거나 다양한 머신러닝 및 자연어 처리 작업을 수행할 수 있습니다. 이 과정에서 임베딩 과정은 텍스트의 의미적인 정보를 보존하도록 설계되어 있고, 벡터 공간에서 가까이 위치한 것들은 의미적으로 유사하다고 합니다. **그러나 임베딩 모델의 성능이 좋지 않으면** 의미적으로 관련된 문서라도 **벡터 공간에서 멀리 위치**하게 됩니다. 이때 임베딩 모델의 성능은 검색 관련 메트릭(MRR, NDCG, Precision@K/Recall@K), 의미적 유사도 메트릭(상관계수, 코사인 유사도), 벤치마크 데이터셋(BEIR, STS), 군집화 품질 메트릭(실루엣 점수, V-measure), 계산 효율성 메트릭(QPS, 임베딩 차원 수) 등으로 측정할 수 있습니다. 저희 팀에서는 서비스화를 위해 처리 속도와 정확도 간의 최적화를 목표로 삼고 오픈소스 모델(dragonkue/BGE-m3-ko, intfloat/multilingual-e5-large-instruct)과 API로 제공하는 유료 모델(solar-embedding-1-large, openai/text-embedding-3-large))을 비교 실험했습니다.

  그리고 현 프로젝트에 최적화 된 임베딩 모델을 찾아서 적용한다고 하더라도
  Vector DB의 선택에 따라 검색 속도, 정확도, 리소스 사용량 등이 달라질 수 있습니다. 따라서, Chroma, FAISS, ElasticSearch, Milvus, Pinecone 등을 살펴보며 Vecor DB를 선택했습니다.

- **Retrieval & Reranker :** Retrieval은 사용자의 질문과 관련된 문서를 검색하는 과정입니다. 이 단계에서의 목표는 신속하게 정확한 문서를 잘 찾아내는 것입니다. Reranker는 리트리버에서 찾아낸 문서들의 순위를 재조정하는 역할을 합니다. 저희 팀은 검색 단계에서 100개를 불러오고 **리랭커로 순위를 재조정해서 모델에 Contexts를 제공하는 초기 전략을 세웠습니다.**

- **Groundedness Checker 및 질문 재작성 :** 복잡한 질문(검색에 필요한 키워드가 많을 때)이 나올 경우 검색 단계에서 질문에 필요한 문서가 없을 수도 있습니다. 그래서 **단계적으로 질문과 답변을 평가**해서 **추가 검색이 필요한 부분을 재검색** 할 수 있도록 Query를 재작성 하는 로직을 추가했습니다. 이 단계에서는 평가 임계값을 몇으로 할지, 평가의 이유를 어떤 기준으로 출력하게 만들지, 질문 재작성의 로직을 어떻게 설정할지 등을 프롬프트 엔지니어링으로 조절하는 실험을 수행했습니다.

- **프롬프트 엔지니어링 + LLM + (답변 결합(Combine Answers)) :** LLM에 Query와 Contexts를 입력하여 사전에 작성한 시스템 프롬프트와 유저 프롬프트를 통해 원하는 답변을 받는 단계입니다. **Groundedness Checker에서 평가 후, 임계값 밑으로 나오면 평가와 질문 재작성 단계를 거치고 출력된 답변들에 대해서 원래 질문의 의도에 맞는 답변을 재결합 시키는 역할도 합니다.** 이때, Input으로 어떤 값을 넣어야 할지 고민이 필요했으며, 적절한 프롬프트 엔지니어링 실험을 통해 원래 질문에 최적화 된 답변을 할 수 있도록 유도하는 작업이 필요했습니다. 또한, o1-mini, gpt-4o, gpt-4o-mini 간의 G-EVAL 성능 테스트를 실험해보기도 했습니다.

G-EVAL 평가

G-EVAL은 ChatGPT를 평가자로 활용하여 생성형 AI 모델의 출력을 평가하는 프레임워크입니다. 이는 G-Eval, a new GPT-4-based NLG evaluation method, has been proposed to improve human alignment (Liu et al., 2023) 에서 기존의 자동 평가 메트릭의 한계를 극복하고자 제안된 방법입니다. ROUGE, BLEU와 같은 메트릭보다 인간 평가자의 판단과 더 일치하는 결과를 보이며 다양한 평가 기준을 유연하게 적용할 수 있다는 장점이 있습니다. 하지만 모델 사용에 따른 비용 발생과 평가의 재현성 및 일관성을 완벽하게 보장하기 어렵다는 단점이 있습니다.

한편으로 RAG 시스템의 성능을 평가할 수 있는
RAGAS 방법론도 있습니다. 답변의 신뢰성, 문맥 관련성, 답변 관련성, 문맥 정밀도 등 RAG 시스템의 전체 파이프라인 평가에 특화되어 있으며 오픈소스 프레임워크로 제공됩니다. 하지만 현 프로젝트의 정량평가 부분에서 G-EVAL이 사용되기 때문에 G-EVAL을 통해 성능 비교 및 최적화를 진행했습니다. 추후에는 RAGAS 평가를 통해 기술적 성능을 최적화한 뒤, G-Eval을 통해 실제 사용자 중심의 평가를 병행하는 하이브리드 평가 방식을 적용할 수도 있을 것 같습니다.

RAGAS vs G-EVAL

	RAGAS	G-EVAL
평가 목적	모델 성능의 세부적인 진단 및 개선	API 결과의 실제 사용 가능성 및 서비스 품질 평가
평가 대상	Retrieval과 Generation 단계를 독립적으로 평가	Retrieval과 Generation 단계를 통합하여 총점으로 평가
평가 지표	4가지 지표(Relevance, Factuality, Coverage, Conciseness) 중심	명확히 정의된 점수 체계(20점 Retrieval, 30점 Generation)
평가 방식	세부적인 정량 지표 + 정성적 평가	OpenAI GPT 모델을 활용한 직관적이고 간단한 평가
적용성	모델 개선을 위한 기술적 평가	실제 사용자 만족도를 중심으로 한 서비스 품질 평가

G-EVAL 평가를 위한 데이터셋 구축은 다음과 같은 과정을 거쳤습니다. 이를 통해 총 100개의 Question과 Groundtruth를 생성하였습니다.

1. metadata의 회사별로(10개 회사) 청킹된 문서를 무작위로 10개씩 가져옵니다.
2. 100개의 랜덤한 문서 중 40개는 쉬운 난이도의 Question과 Groundtruth, 30개는 중간 난이도, 30개는 어려운 난이도로 나눕니다.
3. 쉬운 난이도와 중간 난이도는 1개의 문서만 참고해서 만듭니다.
4. 어려운 난이도의 경우는 2개의 문서를 참고해서 만듭니다.

평가 기준표

Retrieval Evaluation(20점)

#	Evaluation Criteria	Weight	Yes/No
1	Do any of the retrieved contexts show strong similarity to the Ground Truth?	5	
2	Do the retrieved contexts collectively capture essential information from the Ground Truth?	5	
3	Do the retrieved contexts sufficiently address the user's question?	4	
4	Are all retrieved contexts relevant to the Ground Truth or the user's query?	3	
5	Does the combined length and number of retrieved contexts remain reasonable without overwhelming the user with excessive or irrelevant details?	3	

Generation Evaluation(30점)

#	Evaluation Criteria	Weight	Yes/No
1	Is the final answer clearly relevant to the question and reflective of the user's intent?	5	
2	Is the answer factually correct and free from unsupported or inaccurate information?	5	
3	Does the answer include all essential points required by the question and the ground_truth_answer?	5	
4	Is the answer clear and concise, avoiding unnecessary repetition or ambiguity?	5	
5	Is the answer logically structured, consistent with the context, and free of contradictions?	3	
6	Does the answer provide sufficient detail for the question without being excessive?	3	
7	Does the answer provide proper citations or indications of the source when claims or data are referenced?	2	
8	Is the answer presented in a suitable format (list, table, short text, etc.) for the question?	1	
9	Does the answer offer any helpful extra insights or context that enrich the user's understanding (without deviating from factual correctness)?	1	

G-EVAL에 사용된 Retrieval 평가 기준과 Generation 평가 기준표

Multi-Agent 구축

- **전문가 의견 종합 전문가 :** 여러 분야의 리포트(증권사 리포트, 뉴스, 거시경제, 재무제표, 일봉/차트)을 종합하여 하나의 완성도 높은 리포트를 생성하는 역할을 수행합니다. 분산된 정보로부터 고급 추론을 기대하기 때문에 사용된 모델은 o1-mini이며, 각 5개 하위 에이전트는 gpt-4o-mini입니다.

    - **재무재표 전문가 :** 재무제표는 기업의 재무 건정성과 미래 성장 가능성을 평가하는 핵심 도구입니다. 단순한 손익계산서나 대차대조표의 수치를 넘어 각종 재무비율은 기업의 내재가치와 지속가능한 성장 잠재력을 파악하는 데 결정적인 역할을 합니다. 예를 들어 ROE와 영업이익률은 경영 효율성을 보여줍니다. 부채비율과 유동비율은 재무 안정성을 보여주며, 자산 회전율과 매출채권회전율은 자산 활용의 효율성을 나타냅니다.

      그래서 재무재표 에이전트는 특정 기업의 재무제표 데이터를 yfinance 라이브러리를 이용해서 가져오고 각 항목(건전성, 수익성, 성장성, 유동성, 활동성)으로 분류합니다. 이를 통해서 기업의 영업 능력, 자금 조달 구조, 성장성, 수익성 등을 다각도로 분석하여 투자자들이 보다 객관적이고 데이터에 기반한 투자 결정을 내릴 수 있도록 도와줍니다.

- **거시경제 전문가 :** 거시경제 지표를 수집하고 분석하여 한국 주식 시장에 대한 인사이트를 제공하는 에이전트입니다. 거시경제 지표는 기업의 실적과 투자자 심리에 직접적인 영향을 미치는 핵심 요소입니다. 예를 들어, 환율 변동은 수출 기업의 실적과 외국인 투자자금 흐름에 영향을 주며, 금리는 기업의 자금 조달 비용과 투자자들의 자산 배분 결정에 중요한 역할을 합니다. 또한 원자재 가격의 변동은 제조업체의 수익성과 물가에 직접적인 영향을 미칩니다.

  따라서 거시경제 에이전트는 환율, 금리, 원자재 가격 등의 실시간 데이터를 수집합니다. 그리고 거시경제 변수들이 주식 시장에 미치는 복합적인 영향을 LLM이 분석하여 투자자들의 의사결정에 필요한 통찰력을 제공합니다.

- **뉴스 전문가 :** 기업이나 산업에 대한 뉴스는 주가에 즉각적인 영향을 미치는 핵심 요소입니다. 실적 발표, 신제품 출시, 대규모 계약 체결, 경영진 변경 등의 뉴스는 기업 가치 평가에 중요한 정성적 정보를 제공합니다. 특히 최신 뉴스의 감성 분석(긍/부정)은 단기적 주가 움직임을 예측하는 데 도움이 되기도 합니다.

  그래서 뉴스 에이전트는 Google News RSS를 사용하여 특정 기업의 뉴스 검색을 수행하고 각 뉴스 항목의 제목(헤드라인)과 발행일 정보를 추출하여 종합적인 분석을 제공합니다.

- **일봉/차트 전문가 :** 일봉/월봉 차트 분석은 주가의 추세와 모멘텀을 파악하는 핵심 도구입니다. 이동평균선, MACD, RSI, 볼린저밴드 등의 기술적 지표는 주가의 단기적 흐름과 매수/매도 시점을 판단하는데 중요한 참고 자료가 됩니다. 또한 거래량과 가격 변동의 관계 분석은 시장의 심리와 향후 추세를 예측하는데 도움이 됩니다.

  일봉 차트 전문가 에이전트는 한국투자증권 API를 통해 주가 데이터를 수집하고 기술적 분석을 수행하여 투자 전략을 제시합니다. 기술적 분석은 다른 분석(재무제표, 뉴스)과 함께 활용될 때 더욱 유의미한 투자 시그널을 제공할 것입니다. 이를 전문가 의견 종합 전문가에게 기대하는 중요한 점이기도 합니다. 추후 프롬프트 엔지니어링을 통해 이러한 점에 가중치를 크게 둘 수 있을 것입니다.

- **증권사 리포트 전문가 :** 증권사 리포트는 기업에 대한 가장 전문적이고 심층적인 분석을 제공하는 자료입니다. 애널리스트들의 실사와 산업 분석을 바탕으로 한 목표주가와 투자의견은 기관투자자들의 투자 결정에 직접적인 영향을 미칩니다. 특히 리포트에 포함된 실적 전망, 업계 동향, 리스크 요인 등은 기업의 미래 가치를 평가하는 데 핵심적인 정보를 제공합니다.

  증권사 리포트 에이전트는 네이버 기업 해커톤 프로젝트에서 개발된 API를 통해 증권사 리포트 데이터를 가져오고 목표주가, 전문가 투자의견 등을 분석하여 구체적인 투자 전략을 제시합니다. RAG 시스템을 통해 관련성 높은 리포트를 효과적으로 검색하고 분석하여 보다 정확한 투자 인사이트를 도출합니다.

- **리포트 품질 평가 전문가 :** 주식 리포트의 품질을 평가하는 에이전트입니다. 품질의 범위는 float(0-10)이며 임계값(5)을 넘지 못하면 하위 에이전트인 리포트 품질 판단 전문가에게 어떤 에이전트가 부족한 점을 보이고 있는지 판단 요청을 합니다.

  LGAI-EXAONE/EXAONE-3.5-7.8B-Instruct 모델을 활용하여 보고서의 품질을 평가합니다. 평가 방식으로 0~9에 해당하는 토큰의 로짓값을 기반으로 가중치를 부여하여 점수를 평가합니다. 이러한 방법은 고정된 입력에 대해서 항상 동일한 결과값을 반환하므로, 추후에 학습 가능한 구조로 고도화 가능하다는 장점을 가지고 있습니다.

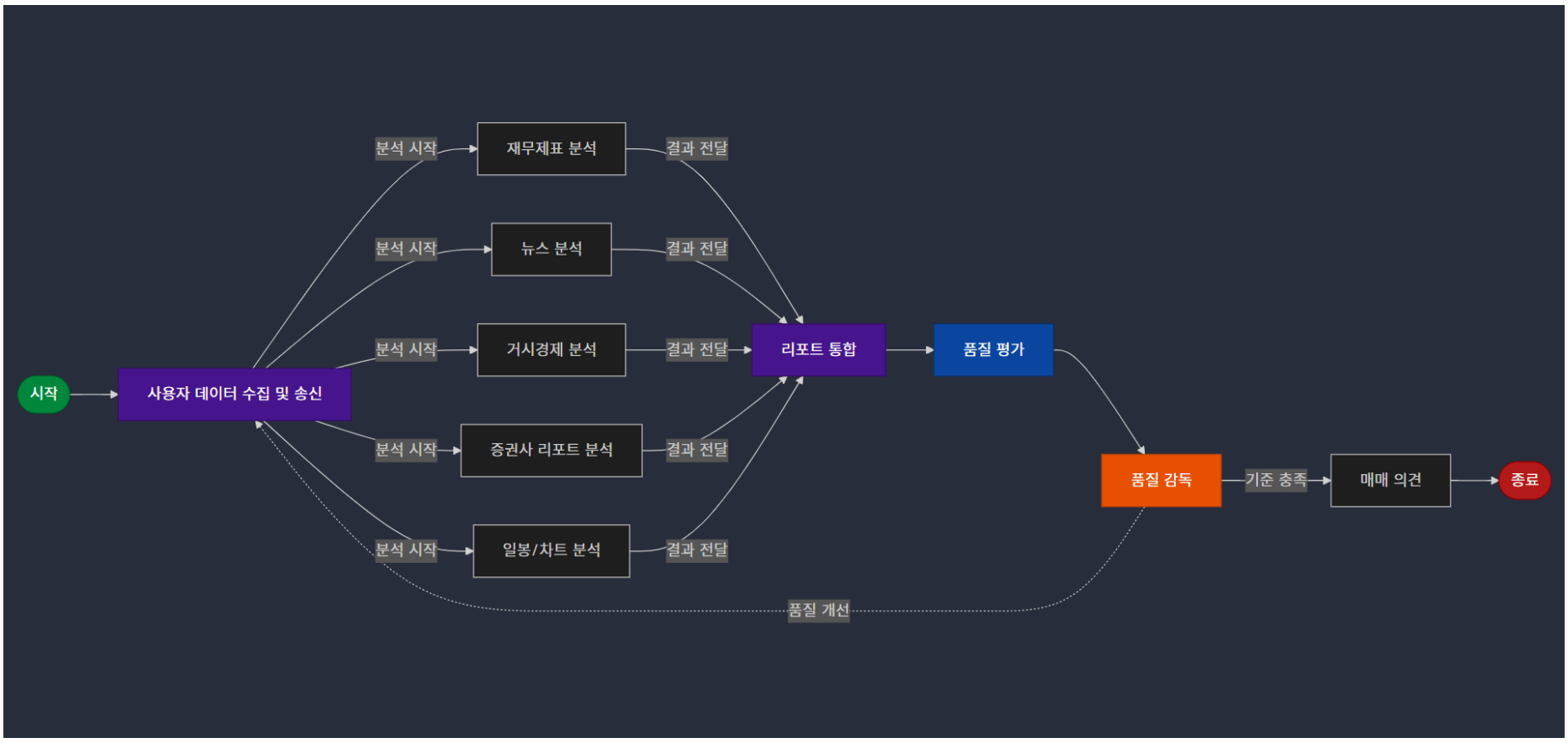
  Encoder 기반의 모델도 탐색하였으나, Encoder 기반의 모델은 position embedding 사이즈가 작고, 학습 데이터셋 역시 구축해야 하기에 적용하지 못하였습니다. 그러나 추후에 유저에게 리포트 평가를 요청하고, 이러한 데이터 기반으로 모델을 학습하는것이 가능하다면 Encoder 구조를 활용함으로써 비용을 절감하는것도 가능하다고 생각합니다.

  - **리포트 품질 판단 전문가 :** 상위 에이전트인 리포트 품질 평가 전문가의 요청에 따라 5개의 에이전트 리포트 품질을 평가합니다. 그리고 진단 결과에 따라 부족한 영역에 맞는 전문 분석 에이전트로 분기하고 재시도 횟수를 관리하여 무한 루프를 방지합니다. 충분한 품질이 확인되거나 재시도 한계를 넘으면 전문가 의견 종합 전문가 노드로 넘어갑니다.

    리포트 품질 판단 전문가의 경우 Intergrated gradients를 활용하는 방식과 에이전트를 활용하는 방식 2가지를 생각해 보았으나, Intergrated gradients를 활용하는 과정에서 n_steps를 증가시킬때 OOM 현상이 발생하는 문제가 있었으므로 에이전트를 활용한 방식을 적용하였습니다.

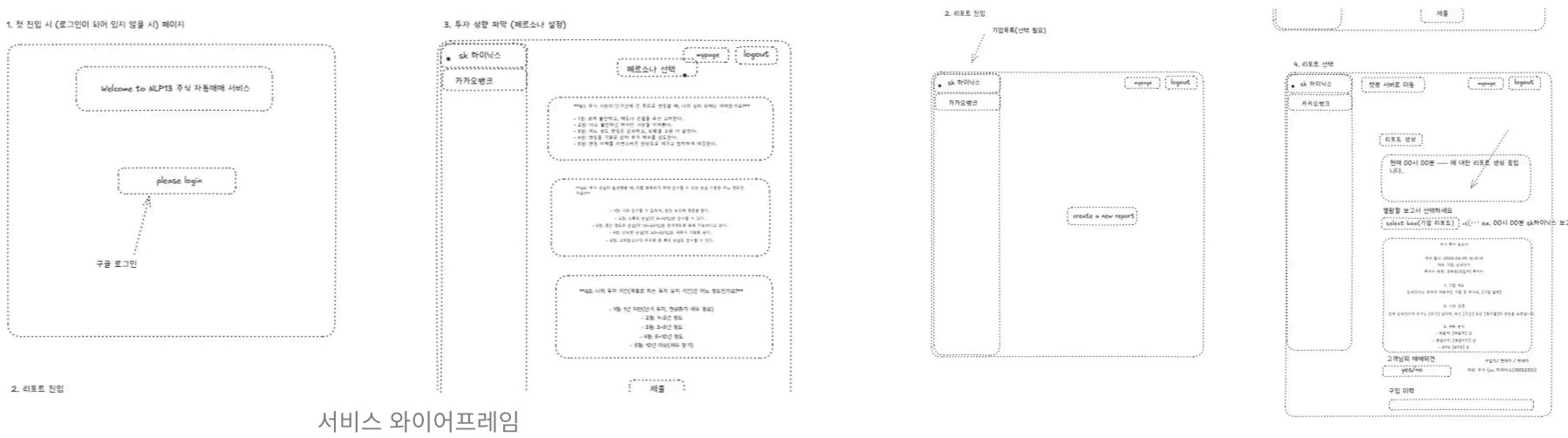
- **매수/매도 의견 전문가 :** 각각의 5개 하위 에이전트로부터 리포트를 받은 전문가 의견 종합 에이전트는 고급 추론을 통해 다양한 변수들을 해석하고 정리된 1개의 리포트를 생성합니다. 이는 품질 평가 에이전트에서 임계값이 5점이 넘으면 매수/매도 의견 에이전트에게 전달이 됩니다. 그리고 고객의 잔고, 손익 등과 함 반영하여 맞춤화 된 매매의견을 제시해주는 역할을 합니다.

Multi-Agent Sketch Version



UI/UX 및 세부 기능 구현

- **사용자 인증:** 구글 로그인 인증 방식을 통해 편리하고 안전한 사용자 인증 제공
- **기업 선택 리스트:** 사용자가 원하는 기업을 쉽게 선택할 수 있는 인터페이스 구현
- **투자 성향 분석 리스트:** 사용자의 투자 성향을 직관적으로 파악할 수 있는 리스트 구성
- **보고서 열람 및 호가창:** 보고서를 열람하고, 실시간 호가 데이터를 확인할 수 있는 UI 구현
- **배치 스케줄러:** 멀티 에이전트 보고서 생성을 위한 배치 스케줄러를 통해 주기적인 자동 보고서 생성



서비스 와이어프레임

서비스 DB 구축

- 주식 자동매매 서비스의 사용자 인증, 보고서 진행 상태 및 열람, 거래 요청, 매매 이력, 유저 데이터, 카카오톡 API 토큰 관리를 위한 PostgreSQL 데이터베이스 구축
- ACID 트랜잭션 지원을 활용하여 데이터 무결성과 안정성 보장
- 확장성과 성능 최적화 고려하여 테이블 설계
- user_id를 외래키로 사용

데이터베이스 테이블 구성

- **user_token (사용자 토큰 관리)**
  - 카카오톡 API의 액세스 토큰 및 리프레시 토큰 저장
  - 토큰 유효기간 만료 시 리프레시 토큰을 활용한 자동 갱신
  - 유니크 제약조건 적용하여 사용자별 토큰 중복 방지
- **trade_requests (거래 요청 관리)**
  - 사용자가 제출한 매수 / 매도 / 홀드 요청 저장
  - 거래 요청 상태 관리
  - 작업 ID(Task ID) 기반 요청 추적 가능
- **trade_history (매매 이력 저장)**
  - 체결된 매매 내역 저장
  - 주식 종목, 거래 가격, 수량, 체결 시간 등의 정보 저장
  - 사용자별 거래 내역 조회 가능

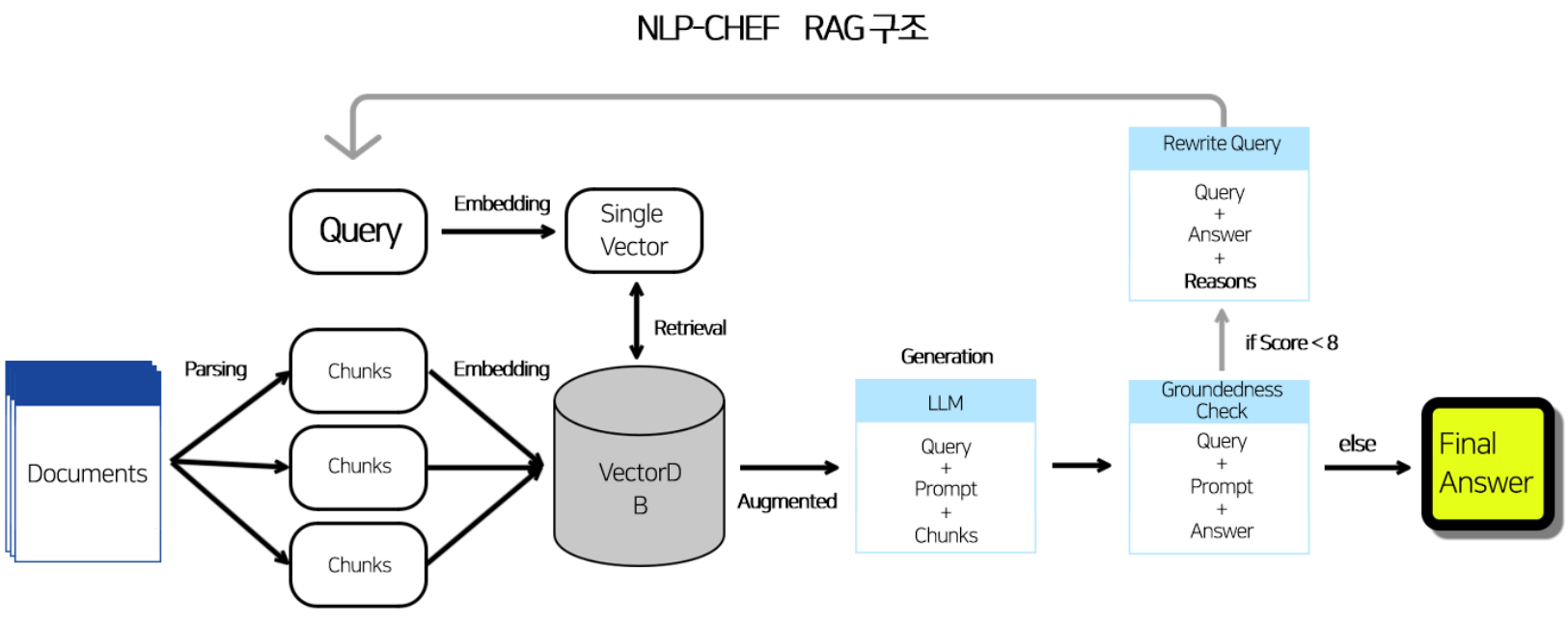
배포

- **도메인 구입 및 등록:** 가비아에서 도메인을 구매한 후, AWS 가상머신에 등록
- **웹 어플리케이션 배포:**
  - Nginx 웹서버를 통해 도메인 바인딩 및 API 프록시를 수행
  - 각 서버에서는 FastAPI 백엔드를 사용해 목적에 맞는 API 제공
- **스케줄러 기반 작업 처리:**
  - 긴 레이턴시 작업을 스케줄러로 안정적으로 처리한 후 API로 배포
- **Docker 기반 서비스 배포:**
  - FAISS와 Elasticsearch를 Docker로 빌드
  - Docker Hub와 Docker Compose를 활용하여 배포

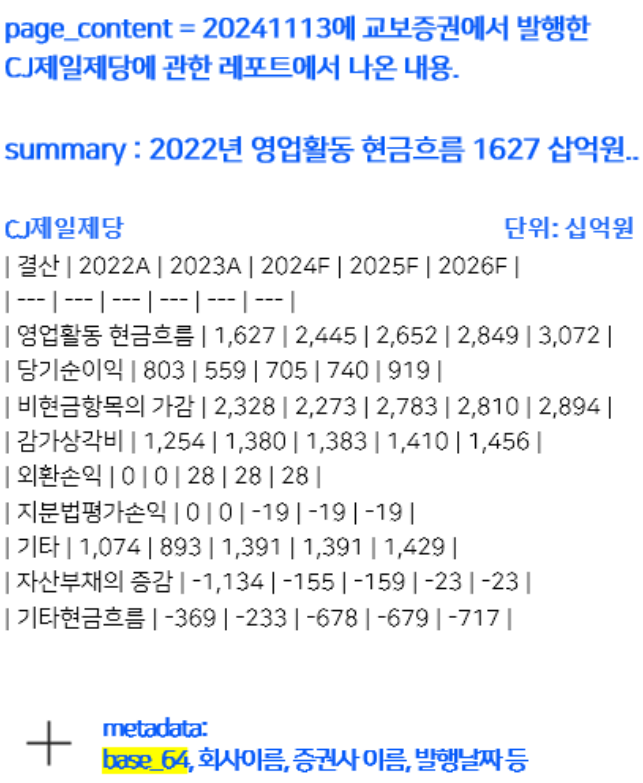
3. 프로젝트 수행 결과

3.1 RAG 파이프라인 구축

최종 RAG 파이프라인



- **파싱 + 청킹 문서 전처리**
  Langchain Document 객체의 page_content는 임베딩 되어 벡터화 될 부분입니다. 이 page_content에는 메타데이터를 통해 {발행날짜} + {증권사 이름} + {회사 이름}을 넣고, 요약한 부분도 추가했습니다. 또한, 메타데이터에 base_64도 추가함으로써 검색된 문서에 base_64가 포함되어 있을 때 테이블과 차트의 이미지를 불러옵니다.



1-4차 파싱 처리를 통해 나온 결과

- **임베딩**

1. sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2			2. dragonkue/BGE-m3-ko			3. Salesforce/SFR-Embedding-2_R		
top_k	MRR	Hit	top_k	MRR	Hit	top_k	MRR	Hit
1	0.1808	0.1808	1	**0.7495**	**0.7495**	1	0.5866	0.5866
3	0.2227	0.2761	3	**0.8025**	**0.8620**	3	0.6580	0.7426
5	0.2329	0.3209	5	**0.8101**	**0.8951**	5	0.6674	0.7826
10	0.2411	0.3837	10	**0.8156**	**0.9358**	10	0.6762	0.8496
50	0.2506	0.5935	50	**0.8175**	**0.9738**	50	0.6807	0.9406
100	0.2522	0.7060	100	**0.8178**	**0.9903**	100	0.6811	0.9655

4. intfloat/multilingual-e5-large-instruct			5. Upstage/solar-embedding-1-large			6. openai/text-embedding-3-large		
top_k	MRR	Hit	top_k	MRR	Hit	top_k	MRR	Hit
1	0.6259	0.6259	1	0.7502	0.7502	1	0.5687	0.5687
3	0.6945	0.7764	3	0.8178	0.8972	3	0.6375	0.7233
5	0.7049	0.8226	5	0.8233	0.9213	5	0.6501	0.7792
10	0.7100	0.8620	10	0.8276	0.9531	10	0.6570	0.8323
50	0.7138	0.9379	50	0.8296	0.9910	50	0.6624	0.9400
100	0.7142	0.9655	100	0.8296	0.9938	100	0.6628	0.9655

임베딩 모델 실험 결과

  - klue-mrc 데이터에서 금융 도메인을 대상으로 1000개로 추린 데이터셋 기반으로 실험
  - mteb leaderboard, 그 외 한국어 임베딩 리더보드들을 참조하여 선정(MiniLM 제외)한 5개의 임베딩 모델을 대상으로 실험을 수행한 결과 가장 좋은 성능올 보인 BGE-m3-ko와 solar-embedding-1-large 모델을 후보로 두었고 각각 dimension tokens이 1024와 4076으로 약 4배 차이나 나면서 성능이 비슷한 점, 검색 시간은 BGE가 더 빠른 점을 고려해서 dragonkue/BGE-m3-ko를 임베딩 모델로 선정
  - 이후 증권사 리포트 기반으로 증강시킨 데이터에 대해서도 좋은 성능

- **벡터 DB(FAISS, ES)**

데이터베이스	설명	장점	단점	서버 필요 여부
Elasticsearch	텍스트 + 벡터 데이터 검색	텍스트와 벡터 데이터를 함께 처리 가능	벡터 검색 성능이 Milvus나 Pinecone에 비해 낮음	로컬 서버, Docker, 또는 클라우드 필요
Faiss	고속 벡터 검색 라이브러리	높은 성능, GPU 가속 지원, 로컬 환경에서 사용 가능	분산 검색 기능 부족, 텍스트 검색 불가능, 대용량 데이터 처리 어려움	서버 불필요 (로컬 실행 가능)
Mivus	벡터 검색 전용 오픈소스 DB	고차원 벡터 검색에서 최고의 성능 제공	텍스트 검색 불가, 서버 설정 필요	로컬 또는 클라우드 서버 필요
pinecone	클라우드 기반 관리형 벡터 데이터베이스	간단한 설정과 관리, 빠른 시작 가능	클라우드 종속, 대규모 데이터 처리 시 비용 증가	자체 서버 불필요 (클라우드 제공)관리, 빠른 시작 가능

Faiss와 ElasticSearch 채택

  - Elasticsearch, Faiss, Milvus, pinecone 4가지 벡터 DB에 대해 사전 조사를 한 후 각각의 장점과 단점을 고려하여 벡터DB로는 Faiss, 키워드 검색 DB로는 Elasticsearch를 선정
  - 두 도구의 강점을 결합한 two-stage 서치를 최종 구현하여 검색의 속도 및 확장성을 동시에 달성
    - 벡터 검색 - 의미적 유사성(Faiss) - Top_k = 100
    - 키워드 검색 - 일치하는 텍스트 (Elasticsearch) Top_k = 15

- **Reranker**
- Reranker(dragonkue/BGE-m3-ko)를 사용함으로써 Latency에서 손해를 봤으나 **G-EVAL 평가에서 2.8% 점수 향상**

	Total_Mean_Score(max=50)	Retrieval_Mean_Score(max=20)	Generation_Mean_Score(max=30)	Total_Latency(100docs/sec)
without_reranker_ver	37.23	14.14	23.09	997
reranker_ver	38.29(+2.8%)	15.49	22.80	1044

동일한 조건에서 리랭커를 적용했을 때 latency가 소폭 증가했으나 정확도가 2.8% 향상

- 질문 재작성 + Groudedness Check 로직 추가 후 G-EVAL 평가에서 **reranker_ver에 비해 9.19% 향상**

	Total_Mean_Score(max=50)	Retrieval_Mean_Score(max=20)	Generation_Mean_Score(max=30)	Total_Latency(100docs/sec)
reranker_ver	38.29	15.49	22.80	1044
enhanced_ver	41.81(+9.19%)	15.42	26.39	1841

리랭커 버전에서 질문 재작성과 Groundedness Check 로직을 도입 후 G-EVAL 점수 9.19% 향상

- 프롬프트 엔지니어링 + LLM
  - RAG 파이프라인에서 **검색 로직(FAISS-ES)**과 **금융 데이터의 표기법을 반영**하여 프롬프트를 설계한 결과 **G-EVAL 점수 4% 추가 상승**
  - 금융 보고서의 표기법(예: “1Q23” “23년 1분기”, “1Q25E” “25년 1분기 예상”, “YoY” “전년비”, “QoQ” “분기”)의 몇 가지 예시를 명시하고 Faiss-ES 검색 로직을 설명한 뒤 질문 재작성 프롬프트를 개선하였습니다.

	Total_Mean_Score	Retrieval_Mean_Score	Generation_Mean_Score	Total_Latency(100docs/sec)
enhanced_ver	41.81	15.42	26.39	1841
Enh_prompted ver	43.48(+4.0%)	16.68	26.80	1910

질문 재작성과 Groundedness Check의 프롬프트를 변경한 후 약 4% 점수 향상

- LLM(o1-mini, 4o, 4o-mini) : 최고 성능이 나온 조건으로 **4o와 o1-mini**를 바꿔서 실험해본 결과 **4o-mini보다 성능이 좋지 않았습니다.** 이에 대한 명확한 이유를 알 수 없어서 한계점으로 남습니다.

	Total_Mean_Score	Retrieval_Mean_Score	Generation_Mean_Score	Total_Latency(100docs/sec)	Token(100docs)	Cost(100docs)
4o-mini	43.48	16.68	26.80	1910	565,762	0.11
4o	41.72	16.56	25.16	2244	681,105	1.76
o1-mini	42.79	16.67	26.12	2687	1,323,500	5.35

동일한 조건으로 모델만 바꿔서 진행해봤지만 예상과는 달리 4o-mini의 성능이 가장 높았습니다.

- 실제 주최 측에서 테스트한 G-EVAL 점수

팀명	Retrieval 점수 (20)	Generation 점수 (30)	합계 (50)	평균 응답시간
NLP-13	19.76	23.5	43.26	29.46

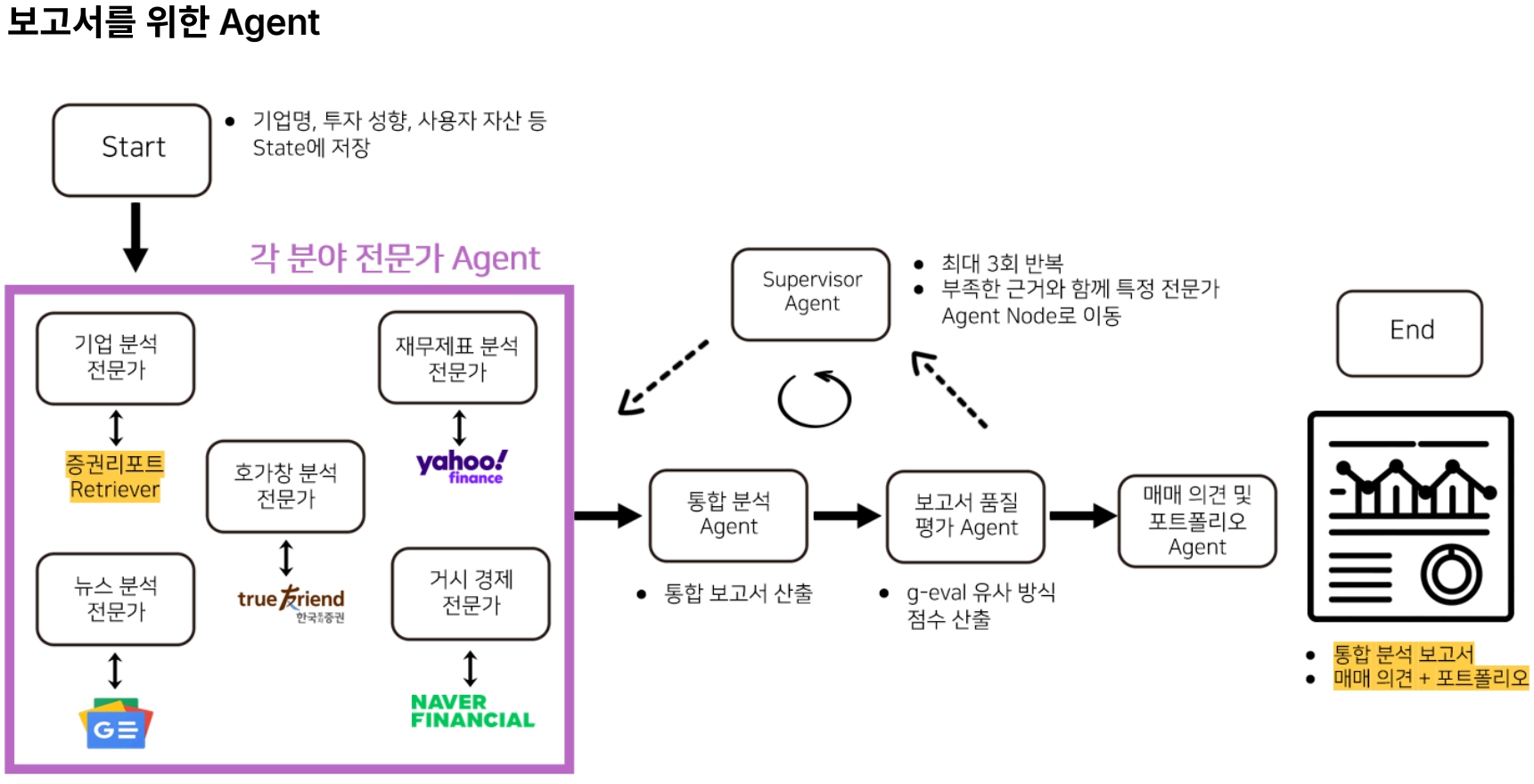
모의 테스트 결과와 실제 테스트 결과는 거의 비슷했습니다. 다만, 총 점수에서는 비슷했지만 Retrieval 점수와 Generation 점수에서 약간의 괴리가 있는 것을 확인할 수 있었습니다.

질문 재작성의 개선된 프롬프트

```
{시스템 프롬프트}:

당신은 하이브리드 검색 시스템을 위한 질문 재작성 전문가입니다.
아래의 검색 시스템을 위한 질문 재작성 지침에 따라 역할을 수행하세요.

검색 시스템은 다음 3단계로 작동합니다.
1. Faiss: 의미적 유사성 기반 검색 (100개 문서 검색)
2. ElasticSearch: 키워드 기반 정확도 검색 (15개 문서 선별)
3. Reranker: 최종 관련성 기반 재정렬

{유저 프롬프트}: ...
```

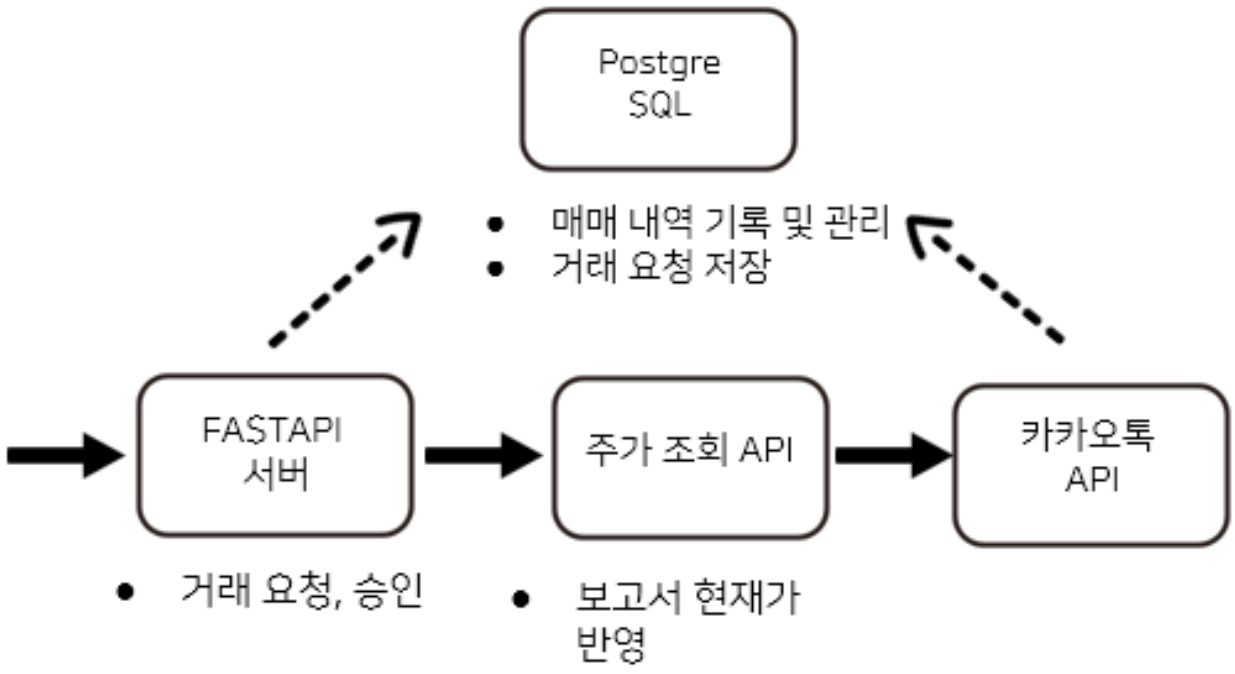
3.2 AI Multi-Agent 주식 매매 관리 서비스

최종 보고서를 위한 Agent



- **기업 분석 Agent**
  - 페르소나: 증권사 레포트 기반으로 주식을 평가하는 전문가
  - 사용한 모델: gpt-4o-mini
  - 출력 보고서 특징: 전문가의 의견이 반영된 품질 좋은 정보 제공
- **재무제표 Agent**
  - 페르소나: 제무제표 기반으로 주식을 평가하는 전문가
  - 사용한 모델: gpt-4o-mini
  - 출력 보고서 특징: 5가지 카테고리의 재무제표 종합 투자 의견과 리스크 요인 파악
- **호가창 Agent**
  - 페르소나: 호가창의 숫자 기반으로 주식을 평가하는 전문가
  - 사용한 모델: gpt-4o-mini
  - 출력 보고서 특징: 일봉/월봉 등 주식 수치 데이터 기반으로 편향 없는 판단 제시
- **뉴스 Agent**
  - 페르소나: 기업과 관련된 뉴스 기반으로 주식을 평가하는 전문가
  - 사용한 모델: gpt-4o-mini
  - 출력 보고서 특징: 최신 뉴스 10개를 기반으로 기업의 긍정적/부정적 측면 제시
- **거시경제 Agent**
  - 페르소나: 거시경제 지표 기반으로 주식시장을 평가하는 전문가
  - 사용한 모델: gpt-4o-mini
  - 출력 보고서 특징: 주식 시장 전반적인 분위기를 분석하는데 강점
- **통합분석 Agent**
  - 페르소나: 5개 Agent의 결과를 받아서 이용자의 회사와 관련된 다양한 정보를 기반으로 고급 추론을 통해 통합 보고서를 작성하는 전문가
  - 사용한 모델: o1-mini
  - 출력 보고서 특징: 한국 주식시장에 영향을 미치는 다양한 변수를 종합해서 사용자의 회사에 영향을 줄 수 있는 부분까지 통합 리포트로 작성
- **보고서 품질평가 Agent**
  - 페르소나: 통합 리포트의 품질을 평가하고 점수를 부여하는 전문가
  - 사용한 모델: deepseek-ai/DeepSeek-R1-Distill-Qwen-7B
- **Supervisor Agent**
  - 페르소나: 보고서 품질 점수를 받아서 임계치(5)보다 낮으면 부족한 근거와 함께 해당 Agent에게 보고서 재요청, 임계치보다 높으면 최종 분석가에게 전달
  - 사용한 모델: gpt-4o-mini
- **매매의견 및 포트폴리오 Agent**
  - 페르소나: 통합 보고서 기반으로 매매의견과 투자 비율을 반환한 이후에, 통합 보고서와 결합하여 최종 포트폴리오를 제시
  - 사용한 모델: o1-mini

매니저 Agent



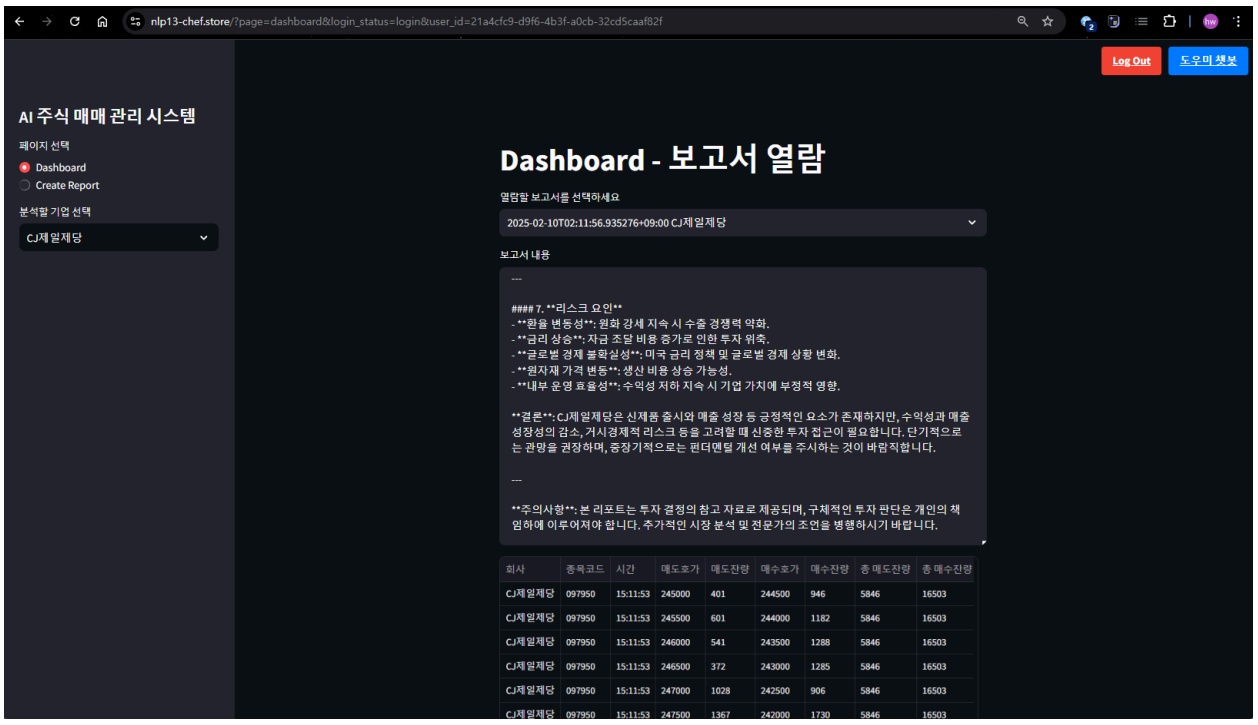
- **매매관리 매니저 구현**
  - 매수, 매도, 홀드 포지션 수신
  - 매매 실행 및 거래 기록 관리 자동화
- **PostgreSQL 데이터베이스 연동**
  - **PostgreSQL**을 활용한 주식 거래 내역 저장 및 관리
  - 거래 이력 기반 의사 결정 보조
- **카카오톡 API 활용한 알림 시스템 구축**
  - 매매 실행 전 확인 메시지 전송
  - 사용자가 거부 시 리포트 재생성 요청 송신

백엔드 연동

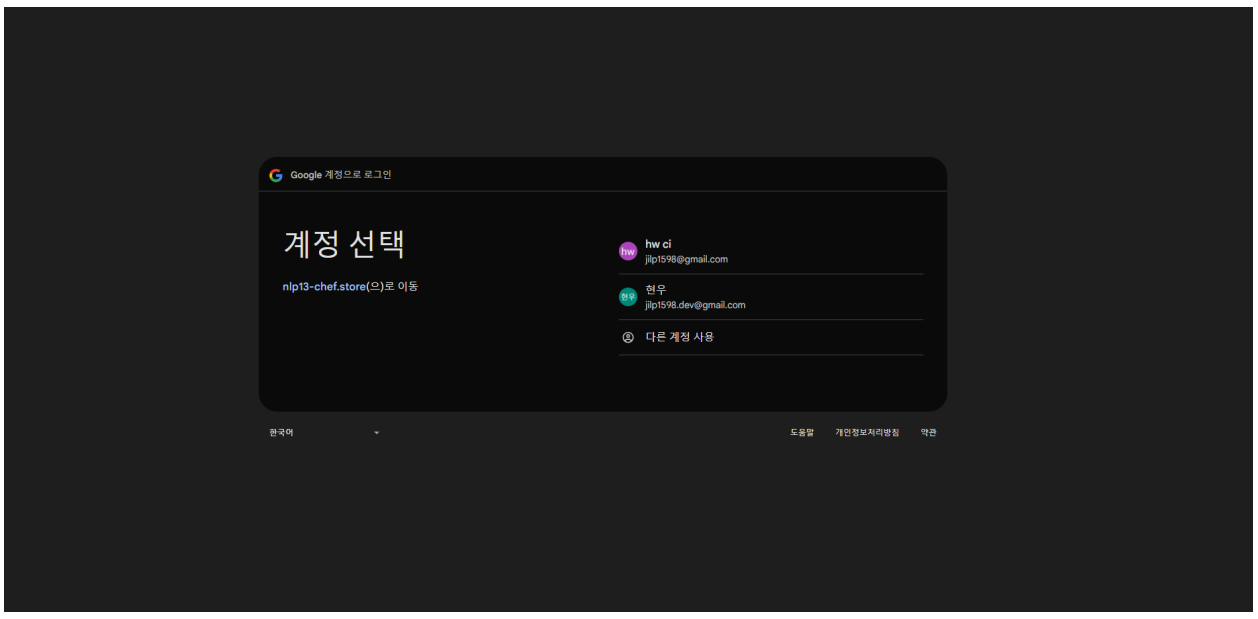
- **매매 실행:** FastAPI 기반 백엔드 서버와 연동하여 매매 실행
- **거래 로그 관리:** PostgreSQL을 활용한 거래 로그 저장 및 관리 API 개발
- **구글 로그인 인증 관리:** FastAPI 기반 백엔드 서버와 연동하여 구글 로그인 인증 관리
- **리포트 관리:** PostgreSQL을 활용하여 리포트 저장 및 상태 관리
- **리포트 생성 자동화:** Cronjob을 활용한 스케줄링 워커 기반 리포트 생성

프론트엔드

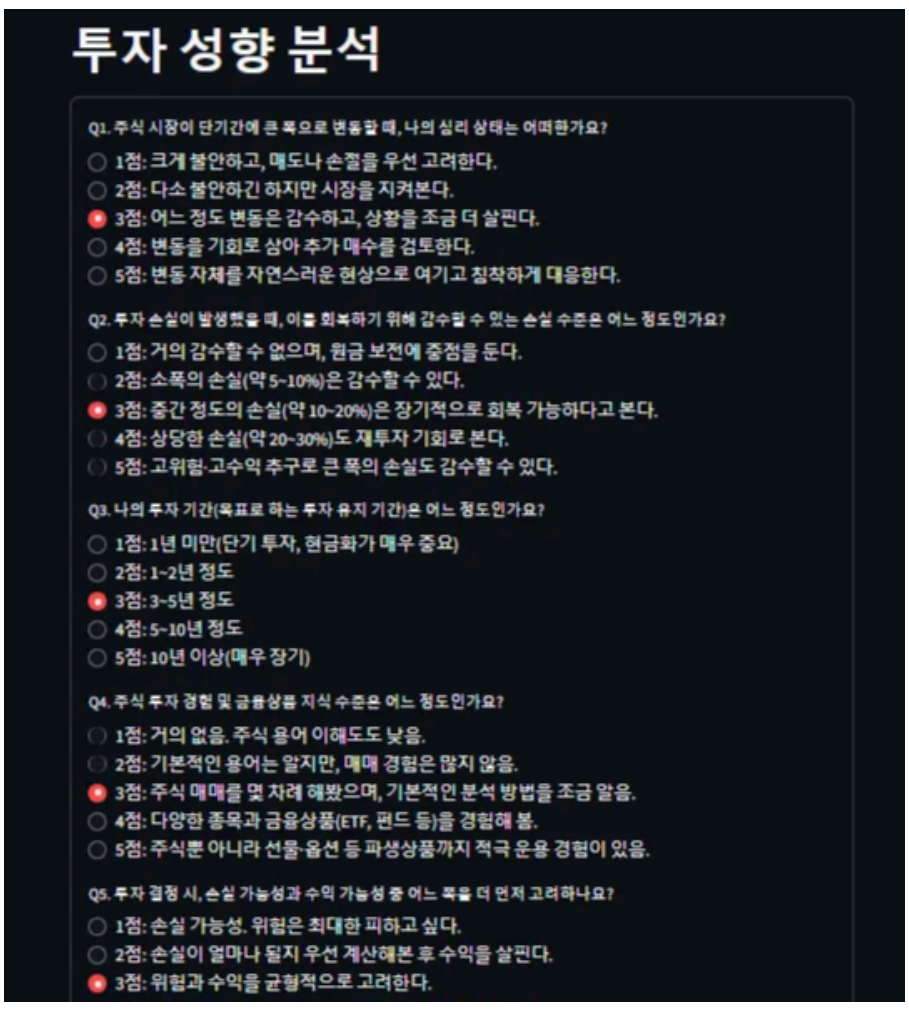
- **Streamlit 기반 UI 구현:** 직관적이고 사용하기 쉬운 인터페이스 제공

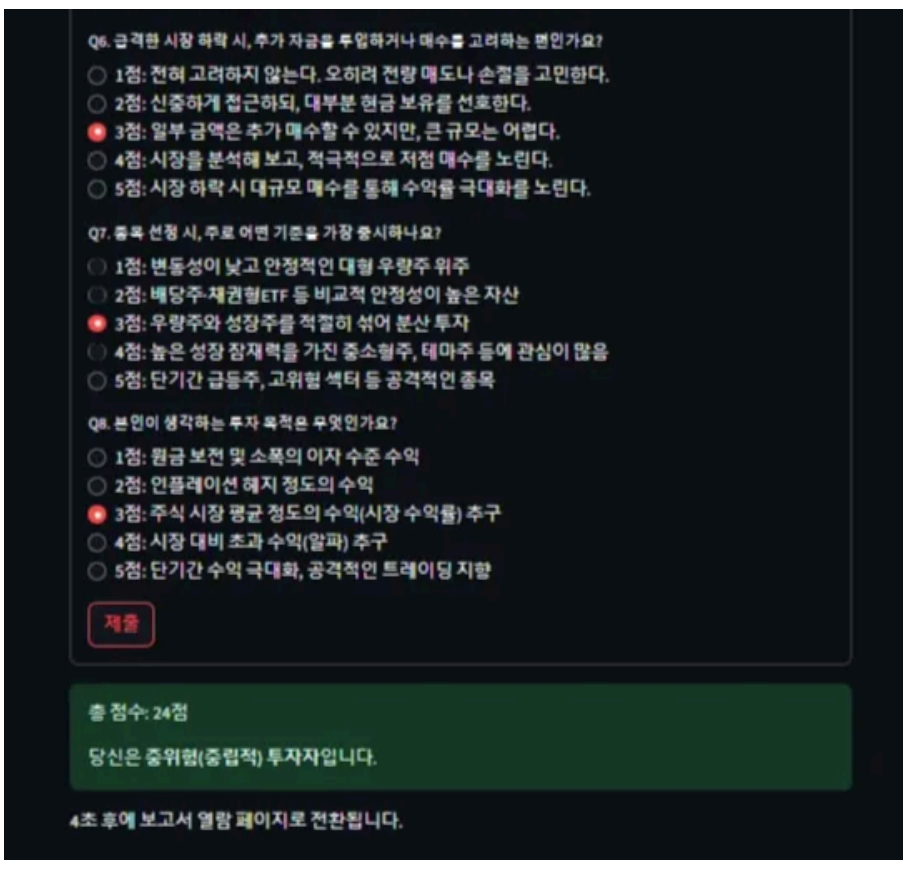


- **구글 로그인 연동:** 간편한 인증 절차로 사용자 접근성 향상

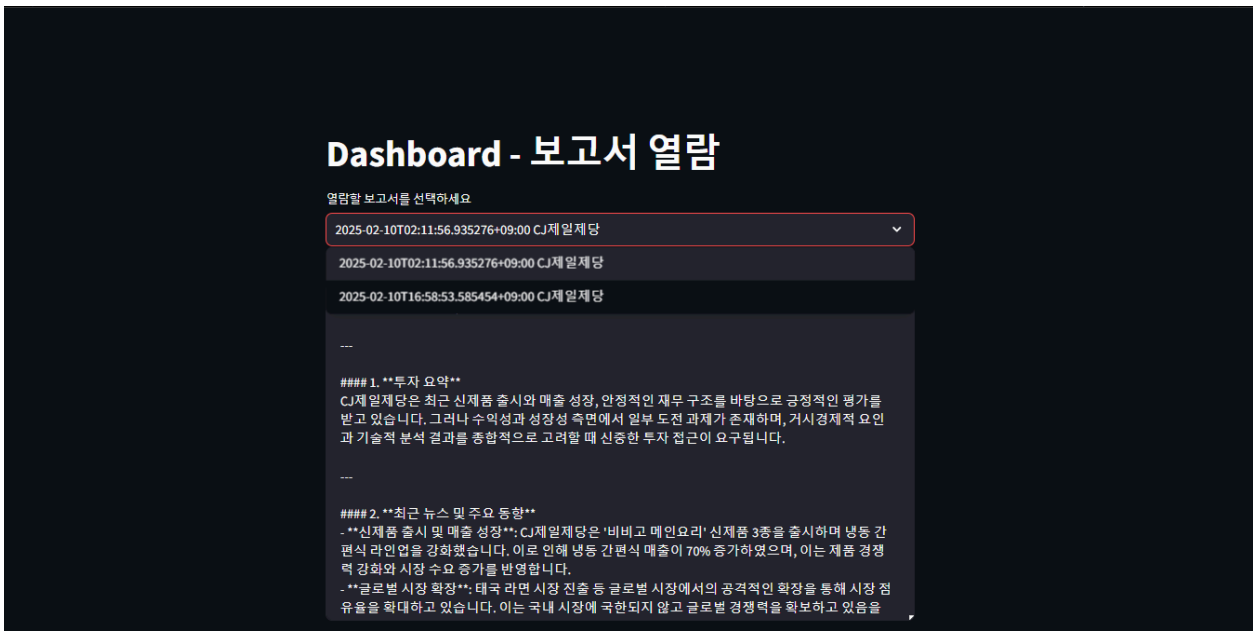


- **사용자 성향 분석 인터페이스 구현:** 개인 맞춤형 분석 결과 확인 가능



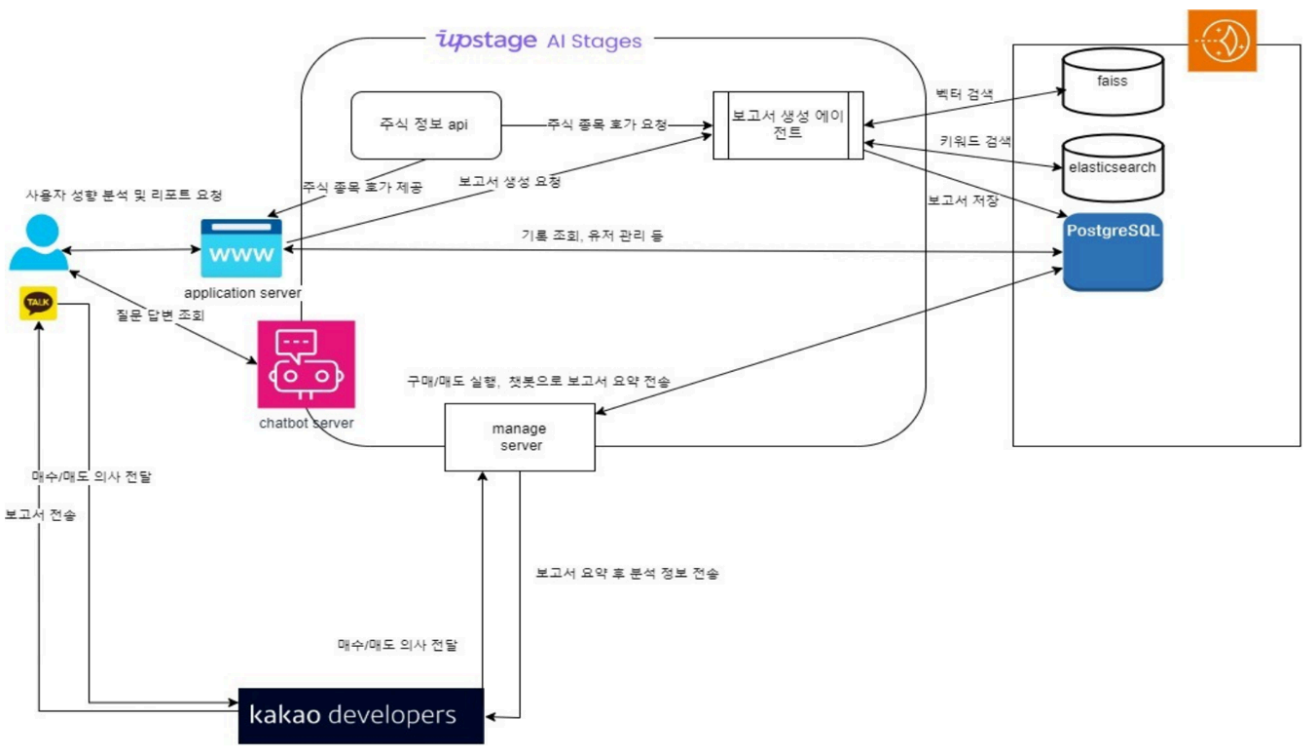


- **리포트 관리:** 저장된 리포트를 손쉽게 선택하고 불러올 수 있는 기능 제공



배포

서비스 아키텍처



- **도메인 구입 및 등록:** 가비아에서 도메인을 구매한 후, AWS 가상머신에 등록
- **웹 어플리케이션 배포:**
  - Web Application 서버에서 Nginx 웹서버를 통해 도메인 바인딩 및 API 프록시를 수행
  - Nginx를 사용해 SSL/TLS 인증서를 적용, HTTPS 프로토콜을 통해 클라이언트와 서버 간의 암호화된 안전한 통신을 구현
  - ai stages의 각 서버들은 FastAPI 백엔드를 사용해 목적에 맞는 API 제공
- **스케줄러 기반 작업 처리:**
  - 긴 레이턴시 작업을 보고서 생성 에이전트 서버의 스케줄러로 안정적으로 처리한 후 API로 배포
- **Docker 기반 서비스 배포:**
  - fastapi로 서버 인터페이스를 구현하여 감싼 Faiss와 Elasticsearch를 Docker로 빌드
  - Docker Hub와 Docker Compose를 활용하여 AWS 상의 VM에 배포

4. 한계점 및 개선 방안

프로젝트의 한계

- 검색 성능이 일관되지 않았습니다. 특히, 하나의 질문에 여러 기업명이 들어간 경우(예: 네이버, 크래프톤, CJ제일제당의 25년도 영억이익...) 답변 품질의 변동이 있었습니다.
- 현재는 에이전트의 워크플로우 직렬화 되어 있기 때문에 비효율적입니다.
- 실제 서비스에는 적합하지 않는 Streamlit 라이브러리를 이용하여 프론트엔드를 구현한 것입니다.
- 서비스 가용성이 적어 다수의 고객을 상대로 서빙하기 적합하지 않습니다.
- 실제 서비스 단계에서의 답변 품질 평가에 쓰이는 G-EVAL로만 성능 비교를 했다는 점입니다.

개선 방안

- 에이전트를 병렬적인 API 호출로 각각 TASK를 수행할 수 있도록 재설계 한다면, Latency가 기존 2-3분에서 30초 내외로 줄어들 것이라 예상합니다.
- 보고서 평가 에이전트 부분에 Tesla V100 32GB 환경에서 7B 모델을 돌리다보니 몇 개의 요청만 들어와도 서버에 무리가 옵니다.
  - 7B → LLM API로 대체
  - Queue를 설정 후 요청에 대해 병렬 작업
- 금융 데이터셋을 구축해서 리트리버를 파인튜닝 한다면 검색 성능이 높아질 것이라고 기대됩니다.
- Groundedness check를 3번 해도 질문에 대한 답변을 찾을 수 없을 때 프론트엔드에서 추천 질문이 나온다면 사용자 만족도가 높아질 것입니다.
- 대부분의 LLM은 영어에 특화되어 있습니다. 그래서 영어로 프롬프트를 작성한 것과 비교 실험을 하는 것도 필요할 것 같습니다.
- 통합 보고서를 유저가 1~10점 사이로 평가할 수 있게 만들고, 유저의 답변 데이터 기반으로 보고서 평가 모델을 학습 시키는 구조를 적용할 수 있을 것 같습니다.
- 실제 서비스 환경에 적합한 프론트엔드 프레임워크와 백엔드 프레임워크를 사용해 구현하여 확장성 있고 안정적인 서비스를 배포 할 수 있을 것 같습니다.
- 대규모 가용성을 지니기 위해 메시지 큐와 쿠버네티스를 활용한 서빙 시스템 구축으로 개선 가능 합니다.
- RAGAS로 더 세밀한 기술적 성능을 비교한 후 서비스적 측면에서의 G-EVAL 성능 비교가 필요했다고 생각합니다.

5. 개인 회고

김현서 : 프로젝트를 진행하면서 아쉬운점은 유저의 데이터 기반으로 개선 가능한 구조를 만들지 못한 부분이 아쉽게 느껴졌습니다. 개인적으로는 발전 가능한 서비스 구조를 만드는 부분이 중요하다고 생각하는데, 현재 아키텍처는 이러한 부분을 만족하지 못하므로 추후에 PRM이나 리포트 평가 로직을 도입하는 부분을 고도화 함으로써 발전시키고 싶습니다. 그리고 모델 서빙 과정에서 자체 모델을 사용할때 사용자의 요청이 몰리는 경우에 GPU가 많이 사용된다는 문제점이 있었습니다. 이러한 문제점을 해결하기 위한 방안으로 쿼리를 큐 단위로 저장한 다음, 배치 단위로 모델에 전달해서 답변을 제시하게 하는 구조를 작성할 예정입니다.

이재룡 : 이번 프로젝트를 통해 보고서 요약, 거래 요청 저장, 사용자 인증 및 승인 요청, 매매 내역 기록 및 관리 기능을 구현하며 FastAPI, PostgreSQL, 카카오톡 API 등을 담당하게 되었습니다. OAuth 토큰 관리가 예상보다 복잡했으며, 보고서 형식을 맞추고 시스템 흐름을 정리하는 과정에서도 어려움이 있었습니다. 향후 거래 승인 프로세스 최적화 및 API 안정성 개선을 진행하고, 자연어 처리(NLP)를 활용한 거래 요청 분석 및 사용자 피드백 자동 처리 기능을 추가하여 더욱 정교한 매매 관리 시스템으로 발전시키고 싶습니다.

이정인 : 팀 프로젝트를 진행하면서 가장 크게 느낀 점은 2가지로 "의사소통의 구체성"과 "과정 관리의 중요성" 이었습니다. 프로젝트를 구성하며 팀원들과 회의를 진행했지만, 제가 제안한 방향을 뒷받침할 충분한 데이터나 사례를 준비하지 못해 설득력이 떨어진 점이 아쉬웠습니다. 시간에 쫓기다 보니 "일단 만들어보자"는 마음으로 서둘러 개발을 시작했는데, 부여 받은 역할에서 결과물이 기대와 달라 다시 처음부터 작업해야 하는 상황 혹은 다른 작업을 수행해야 했던 상황이 많이 아쉬웠습니다.

부족했던 점과 깨달은 것들

- 아이디어 전달 시 "왜 이 방법이 효과적일까?"라는 질문을 스스로 던지지 않은 점
- 중간 점검 없이 장시간 작업하다 보니 방향이 틀어졌을 때 바로잡을 기회를 놓친 점
- 의견을 제대로 전달하지 못한 점

다음 프로젝트에서 실천할 변화 이번 경험을 바탕으로 **3단계 검증 시스템**을 도입하려 합니다.

첫째, 모든 의견 제시 시 최소 1개의 참고 자료(데이터/사례)를 첨부합니다.
둘째, 작업 전 반드시 요구사항을 문서로 정리해 팀원들과 공유할 것입니다.
셋째, 모르는 부분이 생기면 팀원들과 빠르게 소통하며 문제점을 개선할 것 입니다.

아쉬움이 남는 결과물이었지만, 오히려 이 실패가 협업의 본질을 배우는 값진 기회였습니다. 회사 및 팀 프로젝트를 진행할 때는 이번에 얻은 인사이트를 바탕으로 "신뢰받는 협력자"로 성장하고 싶다는 목표가 생겼습니다.

이현풍 :

이번 프로젝트에서는 증권 리포트 기반의 RAG 구축과 MultiAgent 기반 주식 매매 관리 서비스 개발을 중점적으로 진행했습니다.

먼저, RAG 시스템 구축 과정에서 임베딩, Reranker, LLM 모델 등 AI 모델 자체의 중요성보다, 모델 외의 요소들이 얼마나 결정적인 역할을 하는지 깨달았습니다. PDF 파싱과 청킹 작업은 원본 데이터를 체계적으로 분할하고 구조화하는 데 필수적이었으며, 이를 통해 모델이 보다 명확한 정보를 바탕으로 작동할 수 있었습니다. 또한, 프롬프트 설계와 쿼리 재작성 로직은 사용자의 의도를 효과적으로 파악하고 적절한 답변을 도출하는 데 큰 기여를 할 수 있다는 것을 알게되었습니다. 즉, 이러한 전처리 및 후처리 과정이 없었다면 RAG 성능 극대화는 어려웠을 것입니다.

두 번째로, 서비스의 궁극적인 목적에 부합하도록 다양한 목적과 페르소나를 가진 에이전트들을 LangGraph를 활용하여 MultiAgent 시스템으로 구축하였습니다. 각 에이전트가 특정 역할에 집중함으로써 전체 시스템의 효율성과 전문성이 크게 향상되었습니다. LangGraph의 도입은 복잡한 워크플로우를 체계적으로 관리하고, 에이전트 간 협업을 원활하게 만들어 주었습니다. 그러나, 이번 경험을 통해 에이전트 간 통신 프로토콜의 최적화, 전문가 도메인을 활용한 개별 에이전트 성능 향상, 그리고 다양한 시나리오에 대응할 수 있는 기능 확장이 향후 발전 과제로 남는다는 점도 인식하게 되었습니다.

이와 같이, 모델 자체뿐만 아니라 전반적인 시스템 구조와 프로세스 개선의 중요성을 재확인하는 값진 경험이었습니다. 앞으로 이러한 교훈들을 바탕으로 보다 정교하고 효율적인 AI 서비스 개발에 한 걸음 더 다가갈 수 있을 것이라 기대합니다.

임한택 : RAG 시스템을 구축하고 고도화하는 과정에서 문서 파싱과 청킹 작업이 굉장히 중요하다고 느꼈습니다. 그리고 저희 팀이 구축한 검색 시스템과 재무 보고서의 특징(2024년 2분기 -> 2Q24)을 고려하여 프롬프트를 설계했더니 G-EVAL 테스트에서 성능을 20% 가까이 향상시킬 수 있었습니다. 현재 고급 추론 모델인 gpt-o1이나 deepseek R1 모델은 특별한 프롬프트 기술이 필요없다고 하지만, 그보다 성능이 낮은 모델은 굉장히 영향을 많이 받는다 것을 체감할 수 있었습니다. 또한 계층적으로 멀티 에이전트들을 설계하고 구현하는 과정에서 회사 및 사회 조직도와 비슷하다는 생각을 많이 했습니다. 추후 이런점을 참고해서 더 효율적이고 성능 좋은 에이전트 아키텍처를 구현할 수 있을 것 같습니다. 이러 과정을 통해 **프롬프트 엔지니어링의 중요성**과 **멀티 에이전트의 잠재 가능성**이 높다는 것을 다시 한번 확인할 수 있었습니다.

- 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가? 이번 프로젝트에서의 목표는 RAG 시스템을 고도화하는 기술을 습득하는 것과 멀티 에이전트를 설계하고 구현하는 것이었습니다. 목표를 달성하기 위해서 강의와 Langchain 문서를 참했으며, 사전에 간단한 프로젝트를 진행하여 **RAG와 Langchain**을 이해하려고 노력했습니다.

- 나는 어떤 방식으로 모델을 개선했는가? 기존 RAG 시스템에서는 Query가 주어졌을 때 검색을 통해 일방적으로 LLM에 전달되어 답변는 구조로 되어있습니다. 이 과정에서 **검증하는 시스템**과 검증을 통해 나온 이유를 바탕으로 **질문을 재작성하는 로직을 추가**한다면 더 높은 성능을 보일 것이라고 생각했습니다. G-EVAL로 실험한 결과 모델이 개선됐음을 알 수 있었으며, 대회성 프로젝트의 정량 평가의 점수를 올리는 것에도 큰 기여를 했다고 생각합니다.
- 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가? 이전에는 모델 성능을 높일 때 대회에서 지정한 평가 방식이 있었고 리더보드에 따라서 성능 향상의 유무를 결정할 수 있었습니다. 하지만 이번 대회는 자체적으로 데이터셋을 구축하 실험하며 성능 개선을 테스트해야 됐습니다. 이 부분에서 메트릭을 어떻게 설정해야할지, 성능 개선을 보이기 위해서 실험을 어떻게 설계해야 할지 고민이 필요했습니다. 이런 점에서 전과 다르게 **기능별로 개발하는 것과 정량적으로 비교 실험을 하는 것에 더 우선순위를 많이 두었고 실제 성능 향상에 기여할 수 있었습니다.**
- 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가? 실제 사용자 환경을 고려하여 배포까지의 단계를 깊게 생각하지 못했다는 점이 아쉽습니다. 이번 프로젝트에서는 백엔드와 프론트엔드 부분을 기초적인 것으로 구현했지만, **앞으로는 사용자에게 더 나은 만족감을 줄 수 있도록 개발하는 것도 중요한 우선순위를 두어야겠다고 다짐했습니다.**

최현우 : 이번 프로젝트에서 DB 선정 및 임베딩 모델 비교 실험, 서버 인터페이스 설계 및 서비스 아키텍처 구상 및 서빙을 담당했습니다. RAG와 멀티에이전트를 결합한 서비스를 완성해나가는 과정을 보면서 과정은 어렵지 않아 보이지만 실제로 다양한 문제들이 존재하고 이를 해결해 나가는 과정이 쉽지 않음을 알 수 있었습니다. 그리고 높은 latency와 과중한 GPU 메모리 사용량으로 인한 과부하 등 서빙에 어려움이 있었고 이것을 해결하기 위해 고민하는 과정에서 저 스스로 상당히 서빙에 관심이 있음을 느꼈습니다. 그래서 추후 litellm을 통한 빠른추론과 메시지 큐를 통한 과부하 방지, 쿠버네티스를 통한 안정적 리소스 관리 등을 구현해볼 생각입니다.

6. 참고 문헌

1. Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., & Zhu, C. (2023). *G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment. arXiv preprint arXiv:2303.16634.*
2. Déjean, H., Clinchant, S., & Formal, T. (2024). *A thorough comparison of cross-encoders and LLMs for reranking SPLADE*. *arXiv preprint arXiv:2403.10407*.
3. Vatsal, S., & Dubey, H. (2024). *A Survey of Prompt Engineering Methods in Large Language Models for Different NLP Tasks*. *arXiv preprint arXiv:2407.12994*.
4. https://www.elastic.co/elasticsearch
5. https://wikidocs.net/book/14314
6. https://teddylee777.github.io/langchain/rag-tutorial/#google_vignette
7. https://openai.com/index/introducing-structured-outputs-in-the-api/
8. https://docs.smith.langchain.com/