NAVER CONNECT | boostcamp AI Tech

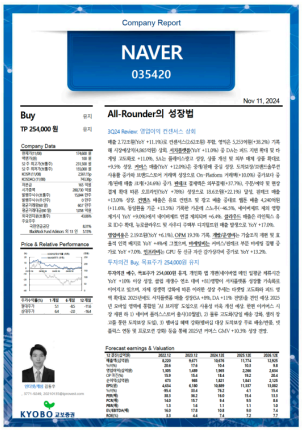
Boostcamp AI Tech Level4 Hackathon Project

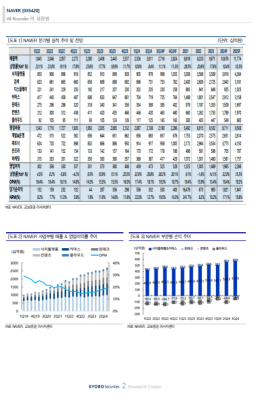
Wrap UP Report

- FinBuddy: 증권/금융 도메인 특화 LLM 챗봇 서비스 -

1. 실습 기간 : 2024. 1. 10(화) 10:00 ~ 2025. 2. 10(월) 19:00
   (부스트캠프 20 ~ 24주차)
2. 프로젝트 주제 : 증권사 자료 기반 주식 LLM 서비스 개발
3. 수행 목표 :
   - PDF 문서로부터 텍스트, 그래프 등 정보의 추출
   - 검색에 적합한 데이터 베이스 구축(VectorDB)
   - 쿼리에 가장 적합한 데이터를 검색해 신뢰할 수 있는 답변을 생성하는 RAG 시스템 구현
   - 프롬프트 엔지니어링 (답변 생성 및 성능 평가)

4. 분석 대상 : 10개 기업에 대한 증권사별 분석 리포트 (IR 평가데이터) PDF 100개





5. 평가 기준 :
   - RAG 성능 – F1, nDCG, MAP 2.
   - 소스코드 품질 – 간결함, 가독성, 모듈화 등
   - 개발 과정과 결과의 효과적인 기록, 문서화

6. 수행 팀 : NLP-09 (뿌이뿌이 모구카)

팀원	김용준	이서현	박수빈	정석현	정유진
캠퍼번호	T7322	T7410	T7338	T7433	T7435

목차

1. 프로젝트 개요

- 증권사의 기업 분석 리포트 PDF 데이터를 Parsing 하여 Vector DB 구축 후 해당 데이터를 기반으로 사용자들이 신뢰할 수 있는 금융 데이터를 편리하게 확인할 수 있는 챗봇 서비스를 제작

2. 프로젝트 팀 구성 및 역할

이름	캠퍼번호	역할
김용준	T7322	PM, PDF 파싱 및 데이터 배포, Agent 스켈레톤 코드 작성, 비동기 Agent 구현, Retrieval, Embedding model, QA모델 평가 코드 개발, BE API 구현, 프로젝트 폴더 구조 설계, 평가 Pipeline 개발, 서비스 파이프라인 개발,
이서현	T7410	Graph/Vector DB 조사, Agent 기반 구축, 검색 및 답변 모델 평가, 프롬프트 엔지니어링, FE 페이지 디자인 및 구현, 모델 평가지표 조사 및 구현
박수빈	T7338	Validation 데이터셋 제작, Retrieval 성능평가, Retrieval 평가 코드 개선, 이미지 처리 Agent 구축, FE API 연동, 회의록 작성, FE 기능 오류 수정
정석현	T7433	답변 모델 구축 및 실험, GPU간 통신환경 구축, 서버 분산처리 실험, FE 연동 및 모듈 개발, 테이블 Agent 및 code execute 모듈 개발
정유진	T7435	PDF 파싱 라이브러리 테스트, Graph DB 조사, Vector DB 조사 및 구축, Agent 기반 구축, 임베딩 모델 조사 및 평가, FE 페이지 디자인 및 구현

3. 프로젝트 수행 절차 및 방법

3-1. PDF 데이터 파싱 및 DB 구축

(1) PDF 데이터 파싱

- PDF를 파싱하는 다양한 파이썬 라이브러리들을 사용했지만 이미지, 표, 텍스트를 모두 완벽하게 잘 추출해내지는 못했음.
- Upstage Document Parser API를 사용하여 PDF 파싱 수행
    - PDF를 여러 부분의 Elements로 분할하기에 파싱에 용이함
    - Output : Category, Content, Coordinates 등
- Category는 Table, Figure, Header, Paragraphs 등 다양하게 표현되고, HTML_content에는 그에 해당하는 마크다운 형태로 담을 수 있음
- 그래프와 표와 같은 형식을 RAG에 활용하기 위해 Multimodal model을 활용하여 해당 요소들에 대한 Summarization 수행

- 테이블은 마크다운형태로 표현하고 이에 대한 Summarization으로, Text로 Summarization으로, Graph는 Graph에 대한 Summarization으로 변환 후 검색에 활용

- 이들을 .csv, .md, 이미지 파일 등 여러 형태로 나눠서 같이 저장

id	type	image_route	dir_route	file_name	page
Data type	String	String	String	String	Int
Example	table, figure	/modules/datas/.	/datas/네이버_교보증권(2024.11.11)	네이버_교보증권(2024.11.11).pdf	1

page	investment	company_name	table	summary
Data type	String	String	String	String
Example	"교보증권(2024.11.11)"	"네이버"	| 현재가(11/08) | 174,600 원 | .	회사 이름과 주식 코드(035420)가 prominently 표시되어 있다

- id : 각 데이터의 고유 식별 기호
- type: 데이터에서 table, figure, graph 등의 이미지 파라미터의 종류 체크
- image_route : 이미지 경로
- dir_route : 이미지가 있는 폴더 경로
- file_name : 이미지가 존재하는 폴더 파일 이름
- page : 이미지가 존재하는 파일의 페이지
- investment : 파일의 증권사 이름
- company_name : 증권사에서 판단한 회사 이름
- table : 마크다운으로 표현된 테이블
- summary : paragraph, table, text에 대한 설명이 담긴 요약(200자 이내)

(2) 모델 평가를 위한 자체 **validation** 데이터셋 구성

- 파싱된 데이터의 Paragraph를 바탕으로 LLM을 이용하여 Query 267 개의 Validation Dataset 제작
- LLM 모델에 프롬프팅과 CoT 를 적용하여 최대한 중복을 피한 형태로 구성하였고, 이를 human eval 하여 데이터셋의 퀄리티를 높임
- 회사명, 리포트 작성 기관, 원문, 질문, 답변으로 구성

(3) **Embedding** 모델 선정

- Naver Cloud Embedding v2 : 네이버 클라우드의 임베딩 모델
- kf-deberta-base : 카카오뱅크의 금융 특화 모델 (도메인 방향성과 일치하여 선택)
- KRUE-v1 : 고려대학교의 bge-m3기반 금융 특화 임베딩 모델
- OpenAi : OpenAi 의 LLM 기반 임베딩 모델

Model	NaverCloud	kf-deberta-base	KURE-v1	OpenAi
Embedding ACC	85.19	78.3	75.23	75.21

- Naver Cloud Embedding 모델이 타 모델 대비 10% 높은 성능을 보임
- DPR retrieval, top-k 5 기준, 자체 제작 validation set 사용

(4) **Vector DB** 구성

- Vector DB
  - 데이터를 수치 벡터로 변환하여 저장
  - 유사성을 기준으로 구조화
  - 빠른 유사도 검색을 지원하여 이미지 및 문서 검색에 적합
- Graph DB
  - 데이터를 노드와 엣지로 구성된 그래프 구조로 저장
  - 데이터 간의 관계를 중점적으로 표현
  - 소셜 네트워크 분석 , 추천 시스템 등에서 유리

=> PDF에서 추출한 텍스트, 이미지, 테이블 데이터를 처리하고 사용자의 쿼리에 적합한 답변이 목적이며 사용자의 query에 적합한 문서를 빠르게 검색하는 것이 목표이므로 Vector DB가 더 적합하다고 판단

- Chroma DB
  - 편리한 사용성과 효율적인 성능을 제공하는 Vector DB 라이브러리
  - 벡터 데이터를 임베딩 하여 저장하고 빠르게 검색할 수 있도록 지원
- Langchain
  - 여러 임베딩 모델을 활용하여 데이터의 벡터 임베딩을 수행
  - Chroma DB 에 벡터들을 저장하여 증권사별 DB 를 구축

현재가(11/08)	174,600 원
액면가(원)	100 원
52 주 최고가(보통주)	231,500 원
52 주 최저가(보통주)	155,000 원
KOSPI (11/08)	2,561.15p
KOSDAQ (11/08)	743.38p
자본금	165 억원
시가총액	280,730 억원
발행주식수(보통주)	15,844 만주
발행주식수(우선주)	0 만주
평균거래량(60일)	60.7 만주
평균거래대금(60일)	1,058 억원
외국인지분(보통주)	43.06%

3-2. Retrieval 모델 구현

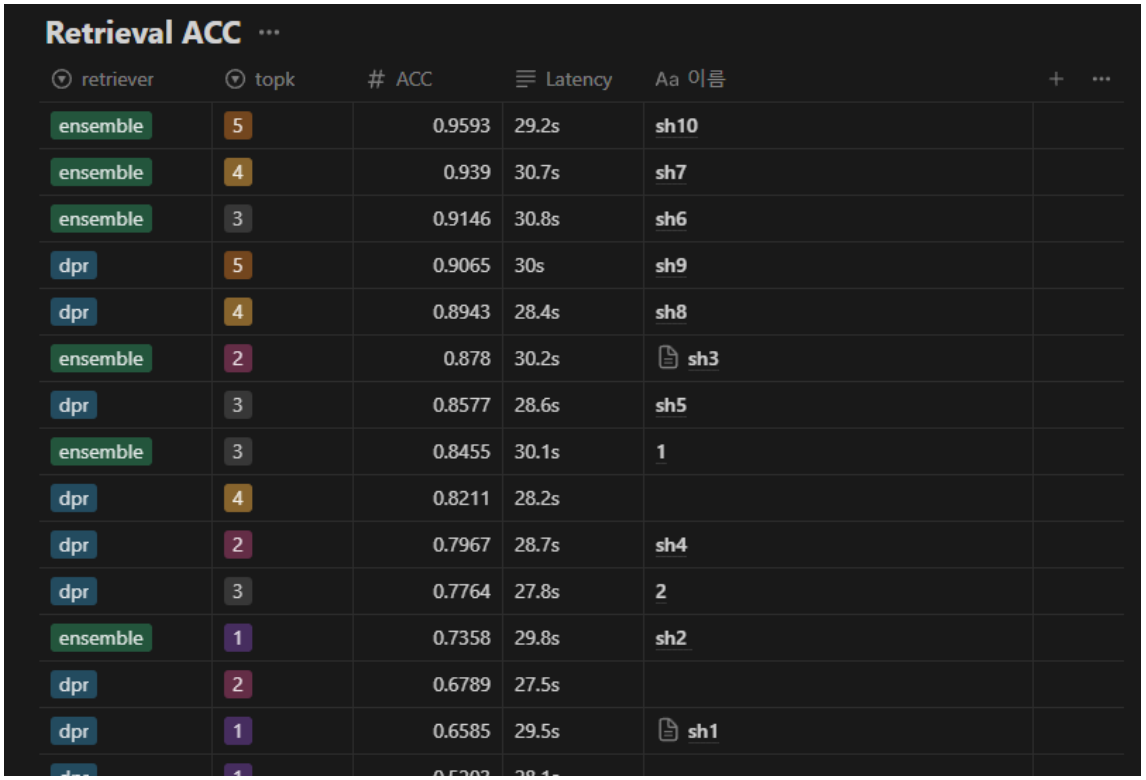
- 파싱된 데이터를 활용하여 구축된 Vector DB를 바탕으로 Dense Passage Retriever 구현
- BM25기반 Sparse Retriever 구현
    - 한국어 형태소 분석기 Kiwi의 Tokenizer를 활용하여 성능 향상
- Dense Retriever과 Sparse Retriever의 Relevant Score을 가중합하여 Ensemble Retriever 구현
- 다량의 문서 (Top-k 20)를 선정하여 LLM이 다시 선택하는 Reranking Retriever도 구현하였으나, 성능 평가가 많이 낮아 이후 프로젝트에 활용하지 않았음

(1) DPR, BM25, Ensemble 방식 비교

- LangChain 에 Retriever 을 이어서 구현
- 질문과 DB 의 summary 를 비교하여 검색
- **DPR** : 신경망 기반의 Dense Embedding을 활용한 검색 기법
- **BM25** : 문서 내 용어의 빈도와 역문서 빈도(IDF)를 기반으로 검색 점수를 계산하는 대표적인 방법
- **Ensemble** : 두 방법론에 대해 서로 찾은 결과에 대해 가중치를 부여하고 우선 순위를 결정

(2) Retrieval 평가

- 평가 방법별, Top k 별, ACC 를 대형 LLM 모델의 프롬프트를 가지고 나온 결과와 비교하여 점수를 비교 및 평가 수행



Retrieval ACC

retriever	topk	# ACC	Latency	Aa 이름
ensemble	5	0.9593	29.2s	sh10
ensemble	4	0.939	30.7s	sh7
ensemble	3	0.9146	30.8s	sh6
dpr	5	0.9065	30s	sh9
dpr	4	0.8943	28.4s	sh8
ensemble	2	0.878	30.2s	sh3
dpr	3	0.8577	28.6s	sh5
ensemble	3	0.8455	30.1s	1
dpr	4	0.8211	28.2s	
dpr	2	0.7967	28.7s	sh4
dpr	3	0.7764	27.8s	2
ensemble	1	0.7358	29.8s	sh2
dpr	2	0.6789	27.5s	
dpr	1	0.6585	29.5s	sh1

3-3. QA model

(1) 모델 분석

- **GPT-4o** : API로 가장 많이 활용하기 좋은 형태를 가지고 있으며 학습 데이터가 매우 풍부하여 주식 도메인에 대한 한정적인 분야에도 잘 대응할 것으로 예상.
- **GPT-4o-mini** : GPT-4o 보단 성능이 조금 떨어지지만 비용적 측면에서 합리적.
- **Exaone 7.8B** : foundation 형태부터 한글 가중치에 대한 학습이 이루어져 한국어 대응에서 가장 뛰어난 성능을 가짐.
- **HyperClova** : 한국어 벤치마크가 뛰어나며 여러 폭넓은 학습이 이루어진 모델.
- **Qwen 32B(4bit)** : 양자화를 통해 v100 GPU에서 가용 가능하며 추론 능력이 뛰어난 대형 모델.

(3) 모델 평가

- 프롬프팅과 Retreival 된 내용을 가지고 응답속도, 각 통계량 별 score 를 환산하여 평가 진행
- 만점은 30점 기준

Model	Time	Mean	std	Min
Exaone 7.8B	5.5s	25.078947	3.709927	10.000000
GPT-4o-mini	1.1s	23.296992	3.813352	14.000000
GPT-4o	1.1s	24.194656	3.470220	12.000000

HyperClova	15s-20s	19.8664	4.577432	0.000000
Qwen 32B	17s-22s	21.373077	4.773160	0.000000

Model	25%	50%(median)	75%	Max
Exaone 7.8B	26.000000	26.000000	27.000000	30.000000
GPT-4o-mini	20.000000	26.000000	26.000000	28.000000
GPT-4o	22.000000	26.000000	26.000000	30.000000
HyperClova	0.000000	26.000000	26.000000	30.000000
Qwen 32B	18.000000	21.000000	26.000000	30.000000

- 점수의 차이를 보았을 때 전반적으로 매우 큰 차이가 나는 형태는 아니지만 최소치 3분위까지의 값에서 Exaone 7.8B의 성능이 좋았음
- GPT-4o와 Exaone 7.8B의 성능이 우수했고, 비용적 측면을 고려하여 무료로 사용 가능한 **Exaone 7.8B** 모델을 최종 모델로 선정

3-4. RAG

(1) RAGAS

RAG 시스템의 성능을 평가하기 위한 프레임워크로, 여러 핵심 지표를 통해 RAG 파이프라인의 성능을 다각도로 평가함.

- Faithfulness (충실성)
  - 생성된 답변이 주어진 컨텍스트에 얼마나 충실한지를 평가
- Answer Relevancy (답변 관련성)
  - 생성된 답변이 질문과 얼마나 관련성이 있는지를 평가
- Context Precision (컨텍스트 정밀도)
  - 검색된 정보의 정확도를 측정
- Context Recall (컨텍스트 재현율)
  - 질문에 대한 답변을 생성하는 데 필요한 컨텍스트 정보를 검색할 수 있는지를 평가

(2) G-Eval

Retrieval 평가 지표 5개, Generation 평가 지표 9개로 구성.

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	

3-5. Multi-Agent

(1) CrewAI

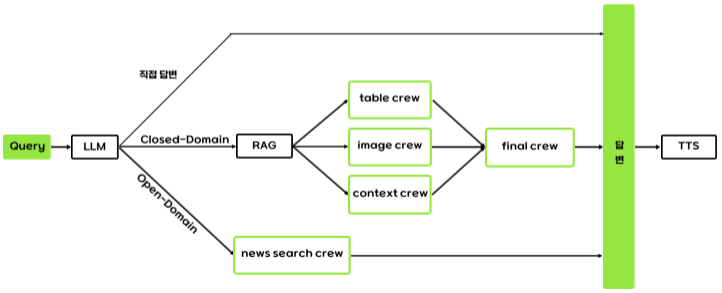
- 자율 AI 에이전트 팀을 구성하고 조율할 수 있는 Multi Agent 프레임워크
- LLM 뿐 만 아니라 기존 Python 함수와의 사용도 용이하게 할 수 있는 형태를 가지고 있어 이번 프로젝트에서 활용

구성요소	설명	주요기능
Crew	전체 AI 팀을 관리하는 조직	AI 에이전트 조율, 워크플로우 관리, 협업 보장, 결과물 생성
Ai Agent	특정 역할을 수행하는 개별 AI	연구, 글쓰기 등의 특정 역할 수행, 도구 활용, 작업 위임 가증 자율적 의사 결정
Process	작업 및 협업 관리 시스템	작업 분배, 협업 패턴 정의, 상호작용 조정, 효율적 실행 보장
Tasks	개별 할당 업무	명확한 목표 설정, 특정 도구 활용, 전체 과정의 일부로 작동, 실행 가능한 결과 제공

Tool	Agent가 사용할 수 있는 Tool	Input 및 Output 명세를 상세하게 작성한 Python 함수 코드. 이를 통해 Agent가 사용자의 질문에 답변을 위해 Tool을 사용할 수 있도록 구성

(2) 모델 비동기 구조

- Agent는 자체적으로 CoT 형태의 구조를 가지고 있고, CrewAI 역시 이러한 Agent들을 Crew 로 사용하여 Sequential하게 처리하는 동기적인 구조를 가지고 있음
- 해당 경우 crew의 개수에 따라 선형적으로 output 시간이 증가하는 문제점을 발견
- 이를 개선하기 위하여 각 Crew별로 async하게 구성하여 table, image, context crew가 비동기적으로 작동한 후, final crew에서 답변을 조합하여 최종 답변을 산출하는 방식을 사용
- 그 결과 기존의 방식보다 약 30~40초 정도의 응답 시간을 감소시켰음



<Fig1. Agent 모델 모식도>

3-6. Product Serving

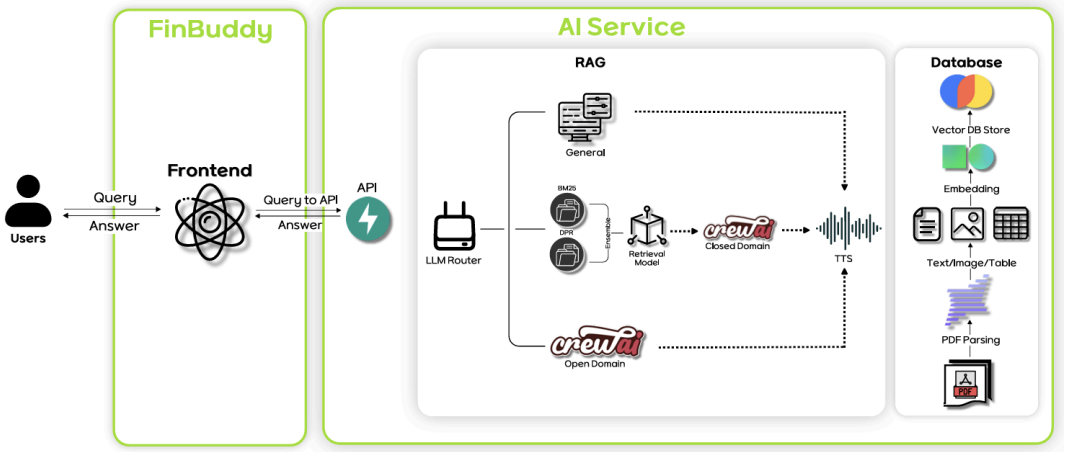
(1) Backend

- FastAPI를 이용하여 API 구성
- Query (GPT 4o mini)
- Open domain query (네이버 뉴스 API를 통해 최신 뉴스 검색후, 같이 답변 제공)
- Closed domain query (주어진 데이터를 이용하여 RAG 후 답변과 참고한 데이터 같이 제공)

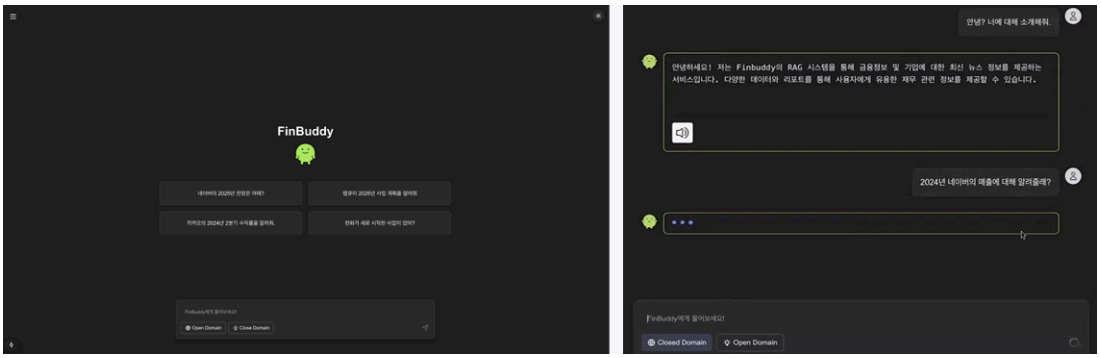
(2) Frontend

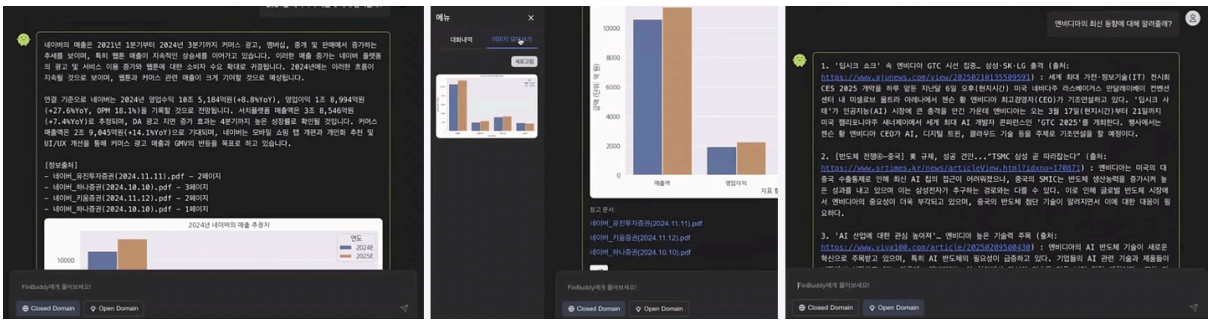
- React, Next.js, TypeScript, Tailwindcss 사용하여 챗봇 웹 페이지 구성성
- axios를 이용하여 API 연동
- 리액트 훅을 이용하여 페이지 변수들 상태관리
- open domain query, closed domain query, query 선택해서 챗봇에게 질문 보내기
- 챗봇에게 응답 받으면 답변이랑 참고한 PDF와 해당되는 페이지, 생성된 이미지 반환
- TTS 연동을 통하여 챗봇의 답변을 소리로 들을 수 있도록 구현
- 오류시 재시도 버튼, 채팅 초기화 기능 구현

4. 프로젝트 수행 결과



<Fig2. Service Architecture>





<Fig3. FinBuddy 웹페이지>

5. 자체 평가 의견

5-1. 발전 방향

1. 사용자 성향에 따른 맞춤 추천
2. 사용자 기능 확장 (회원 가입 및 로그인)
3. 사용자가 직접 데이터 파일 추가
4. 모바일 앱 개발 (리액트 네이티브 등 활용)
5. 채팅 답변 스트리밍 처리

5-2. 협업 및 일정 관리

1. **Git convention**
   - 우리 팀만의 깃 컨벤션을 수립하여 커밋 관리와 브랜치 운영을 체계적으로 관리
   - PR을 통해 피드백을 주고 받으며 코드 이해도를 높이고, 효율적으로 개발 진행
2. **회의 및 문서화**
   - 회의록, 개발 일지, 공유할 내용 등을 git 디스커션과 노션을 통해 꼼꼼하게 문서화
   - 프로젝트 진행 상황을 명확하게 기록하고 공유할 수 있어 팀워크 향상
3. **일정 관리**
   - Jira를 이용하여 프로젝트 일정과 작업을 체계적으로 관리
   - 진행 상황을 실시간으로 추적하여 일정 관리의 효율성을 향상

개인 회고

김용준 (T7322)

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

- 주어진 데이터를 어떻게 활용할지에 대해 가장 먼저 고민하고 파싱 코드를 작성하여 결과물을 공유하여 팀내 피드백을 수용하였음
- 다양한 API(PymuPDF, Upstage Document Parser 등)을 실험한 후 비교하여 가장 결과가 좋은 Tool을 공유하였음
- DB 구축 코드를 작성하여 모든 환경에서 Vector DB를 손쉽게 구축할 수 있도록 하였음
- 서버 환경에 맞는 Unsloth 라이브러리 설치 방법을 연구하고 이를 통해 32GB SRAM의 GPU에서 Qwen32B 모델을 추론할 수 있도록 구현하였고, 설치 과정을 Github Discussion에 공유하였음
- 프로젝트에서 사용하는 Requirements.txt를 통합하여 팀 내 환경설정을 통일하였음
- 팀원들의 프로젝트 수행을 돕기 위해 프로젝트의 폴더 구조를 작성하고 회의에서 코드들의 작동 구조를 자세하게 설명하여 프로젝트의 이해를 높였음
- modules 폴더 내부의 코드들의 기능 및 동작을 전부 README에 기술하여 팀원들의 프로젝트 이해를 높였음
- Agent의 개념을 가장 먼저 공부하여 예시 코드와 함께 Github Discussion을 작성하여 팀 내 개발 역량을 향상시켰음
- 웹 검색 기반 Agent를 구현하여 내부 기반 데이터에 존재하지 않는 내용도 답변할 수 있도록 기능을 추가하였음
- BE API Endpoint를 설계한 뒤 스키마를 공유하여 FE간의 연동 및 페이지 디자인의 방향성을 제시하였음
- FE 페이지의 매핑 실수로 발생한 기능 오류를 수정하고 BE 서버와 알맞는 연동을 통해 PDF파일을 다운받을 수 있도록 구현하였음

2. 나는 어떤 방식으로 모델을 개선했는가?

- 다양한 임베딩 모델과 Retrieval 방법론을 제시한 뒤, 평가 코드를 작성하여 최적의 모델을 선정하는 데에 기여하였음
- 이미지, 테이블, 그래프 데이터를 멀티모달 모델을 통해 Summary로 변환하여 다양한 형태의 데이터를 검색할 수 있도록 구성하였음
- Sparse Retrieval의 토크나이저를 한국어의 특성을 잘 반영하는 토크나이저로 변경시켜 성능을 향상시켰음
- 기업에서 제시한 기준으로 평가하는 G-evaluation을 직접 개발하여 프로젝트의 정량평가에 가장 부합하는 QA모델을 선정하는 데에 기여하였음
- 다양한 프롬프트에 대한 결과를 분석하여 프롬프트 엔지니어링의 방향성을 제시하고, 팀원들이 이해하기 쉽도록 Discussion을 작성하였음
- 동기적으로 수행되던 Multi Agent Pipeline을 비동기적으로 실행되도록 수정하였음

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

- Naver Cloud Embedding 모델을 적극 활용하여 일반적으로 많이 사용되는 bge-m3 또는 llm 임베딩 모델보다 검색 성능을 topk 5 기준 약 10% 이상 향상시켰음
    - 일반적인 리더보드가 모든 도메인에 적용되지 않기에 다양한 모델을 실험하는 것의 중요성을 알게 되었음
- AI가 활용되는 모든 코드의 구조 및 Backend API를 설계하고 기초 코드를 작성하여 팀 내 개발 활동을 도왔음
    - 다양한 예시와 스켈레톤 코드가 개념이나 의견을 효율적으로 전달할 수 있다는 것을 알았음
- 동기적으로 수행되던 Multi Agent Pipeline을 비동기적으로 실행되도록 수정하여 답변 생성

시간을 약 30-40초 단축시켰음
    - 비동기의 개념에 대해 탐구해보는 시간을 가졌고, 오랜 기간 혼자 고민하는 것 보다 더 뛰어난 개발자(문찬국 멘토님)에게 질의하는 것이 문제 해결에 큰 도움이 된다는 것을 알았음
- 라이브러리 내부에서 발생한 오류가 로그가 제대로 표현되지 않아 디버깅에 많은 어려움이 있었는데, 라이브러리의 코드를 직접 탐색하며 디버깅하여 오류의 원인을 제대로 알 수 있었음
    - 단순히 라이브러리를 사용하는 것이 아닌 내부 동작을 살펴보는 것이 타인이 작성한 코드를 이해하는 역량을 많이 기를 수 있으며, 디버깅을 효율적으로 진행할 수 있다는 것을 알았음
- 직접 제작한 평가 데이터를 통한 내부 평가는 높았으나, 이후 기업 해커톤 결과에서 성능이 높지 않았다는 의견을 통해 평가 데이터의 신뢰성 확보가 중요함을 알았음

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?
- 기능을 하나씩 작성하는 것이 아닌 틀을 먼저 작성한 뒤 채워가는 개발 방법을 시도하였음
    - 이를 통해 개발 방향성을 놓치지 않고 하나의 목표를 향해 개발할 수 있게 되었음
- PDF와 같은 이미지 문서를 처리하는 방법에 대해 자세히 조사하였음
    - 다양한 RAG 구조에 대해 알 수 있었고, 이미지 형태의 문서를 처리하는 다양한 방법론에 대해 깊게 탐구할 수 있었음
- 처음 접하는 개념(CrewAI)를 배울 때, 블로그나 유튜브 글이 아닌 공식 문서를 많이 참고하여 학습하였음
    - 영어가 어렵더라도 공식 문서를 살펴보는 것이 가장 학습에 도움이 된다는 것을 알았음
- Github Discussion을 적극적으로 활용하여 개발 흐름 및 코드의 동작을 자세히 설명하였음
    - 개발 일지나 개념 설명 등을 직접 작성하여 공유하는 것이 팀의 개발 활동에 많은 도움이 된다는 것을 알았음
- Github PR을 통해 Main Repository를 안정적으로 유지하는 방법을 습득하였음
    - Github의 사용을 더 효율적으로 할 수 있게 되었고 협업을 위한 프로그래밍에 대해 더 깊게 알아가는 계기가 되었음
- Frontend 및 Backend, Torch 등 다양한 환경을 세팅할 때 버전 충돌 오류가 많았으며, Requirements.txt를 작성할 때 자세한 버전을 명시하였고, 라이브러리 설치 과정을 공유하였음
    - 환경 설정이 잘못되면 많은 과정을 돌아갈 수 있기에 섬세하게 기록하는 것의 중요성을 깊게 이해하였음
- 독립적으로 실행되는 코드를 비동기로 실행되도록 리팩토링하였음
    - 비동기로 실행하는 것이 시간이 많이 소요되는 작업의 Latency를 효과적으로 줄일 수 있음을 알았음
- Agent를 구현하며 무수한 프롬프트 엔지니어링을 수행하였음
    - 자세하고 정확한 프롬프트 작성의 중요성을 알았고, 명확한 명세를 제공하는 것이 까다롭고 많은 시행착오가 필요한 작업임을 깨달았음

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?
- 내부에서 진행한 평가 결과와 실제 기업에서의 정량평가 결과가 차이가 존재하였음
    - 신뢰성 있는 평가 데이터셋을 구축하는 것이 많이 중요하다는 것을 깨달았음
- CrewAI를 통한 Agent는 Text-streaming을 지원하지 않아 텍스트 생성의 시간이 체감적으로 더 길게 소요되었음
    - 다양한 Agent Tool을 꼼꼼히 비교하지 못해 서비스의 퀄리티가 떨어졌다는 부분이 아쉬웠음
- 코드의 오류가 많아 무의미하게 소비된 API Credit이 많음
    - 코드 검증 과정을 좀 더 신경써 무의미하게 소비되는 API Call을 최소한으로 줄일 것임
- 매 시간마다 팀원들에게 Task를 적절히 맡기지 못해 인적자원의 낭비가 존재하였음

    - PM으로서 팀원들의 강점을 완벽히 활용하지 못한 점이 아쉬웠음
- 기획 단계에서 더 다양한 아이디어를 구현 방법과 함께 제시하지 못하였음
    - 기획 이후 개발된 서비스가 타 서비스와 비교해 큰 강점을 갖지 못하는 것 같다는 생각이 많이 들어 아쉬웠음
    - 사용자의 불편함을 캐치하고 분석하는 능력이 많이 부족하다 느꼈고, 이를 꼭 보완해야 한다고 통감하였음
- 프론트 페이지를 말끔하게 구성하지 못해 시각적으로 서비스의 질이 떨어져보이는 단점이 존재했음
    - 프론트엔드를 구성해본 적이 없어 사용자를 최우선으로 고려하는 관점을 갖고 개발하는 역량이 부족하였음

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?
- AI모델을 활용한 서비스 개발에는 내부 평가와 실제 서비스에서의 퍼포먼스가 다를 수 있으며, 이 사이의 간격을 최소화하는 것이 중요하다는 것을 알았음
    - 다음 프로젝트에서는 사용자의 유즈케이스를 더 꼼꼼히 분석하여 실제 환경과 동일한 평가를 구현하도록 신경쓸 것임
- Agent 툴에 대한 기초 지식이 아직 부족함
    - AutoGen, Langchain 등 다양한 Agent 툴에 대한 장단을 확실히 분석하여 서비스의 질을 올릴 수 있도록 노력할 것임
- 이번 프로젝트에서는 팀원의 강점을 면밀히 살펴보지 못하였음
    - 팀원의 역량을 정확히 파악하는 데에 신경을 기울여 리소스가 최대한 낭비되지 않도록 노력할 것임
- 아이디어에 대해 제시를 할 때에는 팀원들이 확실히 이해할 수 있도록 신경써야함
    - Figma와 같은 디자인 툴을 배워 본인의 아이디어를 제대로 설명하는 역량을 기를 것임
- 프론트에 대한 기초 지식이 부족함
    - 모든 과정에서 팀원들이 겪는 어려움을 해소할 수 있도록 다양한 도메인에서의 기초 지식을 갖춰 프로젝트 수행 과정을 매끄럽게 진행되도록 도울 것임

박수빈 (T7338)

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?
- Validation 데이터셋 제작, Retrieval 성능평가, Retrieval 평가 코드 개선, 이미지 처리 Agent 구축, FE API 연동, 회의록 작성, FE 기능 오류 수정

2. 나는 어떤 방식으로 모델을 개선했는가?
- 모델의 Validation 지표를 확인해보기 위해 GPT 4o-mini로 평가 데이터를 제작하여 성능이 좋은 모델을 선택할 때 도움을 주었다.
- 성능이 좋은은 Retriever 선택을 위해 BM25, DPR, 앙상블 리트리버를 topk 1 ~ 5까지 바꿔보며 ACC와 Latency를 기록했다.
- 사용자에게 더 좋은 성능의 답변을 제공하기 위해 각각의 역할이 있는 Multi Agent를 구축하는 과정에서 이미지 처리하는 Agent의 role, gole, description을 바꿔가며 성능을 개선했다.
- FE를 리액트로 구현하는 과정에서 의존성을 고려하여 환경 세팅을 하였고, 팀원에게 세팅법을 전달하였다. FE와 BE API 연동을 하고 챗봇의 답변을 사용자가 보기 편하도록 파싱하여 글, 이미지, 참고한 문서를 따로 보여주었고 시각장애인분들을 위하여 TTS로도 제공하였다. FE 기능 부분에서 오류가 생겼을 경우 해결하였다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

- Validation 데이터셋을 LLM을 이용하여 제작하고 Retriever의 성능 평가를 진행하였다. 이미지 처리를 하는 Agent의 성능을 개선하였다. FE를 구현하기 위한 기반을 만들어주었다.
- React를 처음 써보았는데, 이전에 Vue.js를 써봤어서 그런지 많이 생소하진 않았고 오랜만에 웹 페이지 개발을 하여 하나하나 구현할 때 뿌듯했다.
- 피어세션, 멘토링, 구현한 기능들에 대한 기록을 Github Discussion에 기록하고 팀원들에게 공유하여 참고할 수 있도록 하였다.
- CrewAI를 활용하여 Multi Agent의 성능 개선을 해보며 AI Agent 구현의 큰 흐름을 알게 되었다.
- 주로 BE 개발을 맡아했어서 잘 몰랐는데, BE에서 FE로 데이터를 넘겨줄 때 FE가 데이터를 쓰기 쉽도록 잘 쪼개어 주는 것이 중요하다는 것을 느꼈다.

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 처음 사용해보는 React를 사용하여 웹페이지를 구성해보았고, React Hook과 Props를 이용하여 페이지에서 쓰는 변수의 상태관리를 해보았다. 앞으로 웹 페이지를 만드는 경우, 더 공부하여 계속 사용할 것 같다.
- AI Agent에 관심이 있었는데, 이번 프로젝트를 계기로 구현해보고 사용해보면서 코드가 돌아가는 큰 흐름을 알게 되었다.
- 회의록과 멘토링, 구현한 기능들에 대한 기록을 팀 노션으로만 하다가 Github Discussion을 처음 써봤는데 비슷한 주제에 대해 글을 묶어서 쓸 수 있어서 팀원들끼리 정보를 공유하기가 편했다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- React가 익숙하지 않아서, 짧은 시간안에 더 많은 기능을 구현하지 못하여 아쉬웠다.
- BE 부분 또한 맡아 해보고 싶었으나, FE API 연동을 하는 것에 시간을 많이 쏟아 맡지 못해 아쉬웠다.
- 클라우드 서버에 띄우지 못하여 사용자가 편히 이용할 수 있도록 하지 않은 게 아쉬웠다.
- DB를 연결하여 모델을 통해 생성된 이미지를 저장하면 편했을텐데, 심볼릭링크만을 이용하여 저장공간을 임시로 이어놓은 점이 아쉬웠다.

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 프로젝트에 시간을 더 쏟을 것
- 팀원과의 소통이 더 원활하도록 문서를 작성하고 전달해줄 것

이서현 (T7410)

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

이번 프로젝트에서 나는 RAG 기반 AI 챗봇을 개발하며, Agentic Workflow의 설계 및 최적화와 RAG 성능 평가 및 개선을 주요 학습 목표로 삼았다.

- LangChain & ChromaDB 학습 및 활용: RAG 시스템의 핵심인 벡터 DB 구축을 위해 ChromaDB를 연구하고, LangChain을 활용하여 문서 임베딩 및 검색 프로세스를 설계했다.
- CrewAI를 활용한 다중 에이전트 시스템 구축: CrewAI로 Router 및 각 역할별 Agent를 구현했다.

- 평가 지표 조사 및 성능 분석: RAG 성능을 객관적으로 평가하기 위해 기존 평가 방식들을 조사하고, 이를 기반으로 평가 프레임워크를 구축하여 모델 성능을 모니터링했다.

2. 나는 어떤 방식으로 모델을 개선했는가?

- 검색 성능 개선: RAG 검색 결과의 품질을 높이기 위해 벡터 임베딩 방식과 검색 전략을 최적화하고, top-k 검색 개수를 조정하여 최적의 결과를 찾았다.
- 프롬프트 엔지니어링: LLM의 응답 품질을 향상시키기 위해 다양한 프롬프트 실험을 수행하며, CrewAI 내 agent 간의 협업 방식도 조정하여 보다 자연스러운 응답을 생성하도록 개선했다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

- 효율적인 정보 탐색 제공: 기존에는 개별 증권사 리포트를 다운로드하고 하나하나 찾아봐야 했던 정보를, 질문만으로 빠르게 찾을 수 있도록 했다.
- 검색 정확도 향상: 최적화된 RAG 시스템을 통해 불필요한 정보 노이즈를 줄이고, 질문과 높은 연관성을 가지는 데이터를 우선적으로 제공했다.
- Workflow 설계의 중요성 인식: 다중 agent 기반 시스템에서는 단순히 개별 agent의 성능뿐만 아니라, agent 간 협업 방식이 최종 성능을 결정짓는 중요한 요소임을 깨달았다.

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

이전 프로젝트들과 비교했을 때, 이번에는 Agentic Workflow를 적극적으로 활용했다는 점이 가장 큰 차이였다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- RAG 검색의 한계: RAG 시스템이 검색된 문서 내에서만 답변을 생성하기 때문에, 검색된 정보 자체가 부족하면 답변의 품질도 낮아지는 문제가 발생했다.
- 데이터 업데이트 문제: 금융 데이터는 최신성이 중요한데, 데이터베이스의 업데이트 주기에 따라 최신 정보를 반영하기 어려운 한계가 있었다.

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 실시간 데이터 반영: 최신 금융 정보를 반영하기 위해, API 기반 실시간 데이터 수집 기능을 추가하는 방안을 검도할 예정이다.
- Agent Collaboration 최적화: CrewAI와 같은 Multi-Agent Framework를 사용할 때, Agent 간 협력 구조를 더 정교하게 설계하여 효율성을 높이고자 한다.

정석현 (T7433)

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

- PDF 형태의 데이터 구조에 대해서 Parsing key 들에 대해서 Retrieve 하기 좋은 형태 구축하기 위해 도메인별 방법론들에 대한 조사와 실제 서비스에 사용되는 API들을 살펴보고 이후 해당 구조를 참고하여 평가지표 방식을 탐구하고 서비스 도메인에서 좀 더 사용하기 좋은 형태로 바꾸기 위하여 코드 레벨에서 테스트하였습니다.

- 여러 로컬 LM모델, API LLM모델, VLLM 모델들을 서로 비교하고 특정 도메인내에서의

서비스에서의 장점인 모델을 찾기 위하여 한정된 서버, GPU, 한정된 라이브러리 내(cuda 버전)에서 모델을 불러오는 방법 및 대형 모델 서빙을 하기 위해 멀티 GPU 서버 환경에서의 분산 처리 방법에 대한 학습 및 테스트를 진행하였습니다.

- Agent를 구현하기 위해 Agent 관련 문서를 읽고 어떤 방식으로 동작하는지 crewAI 의 docs , 다른 팀원의 자료를 팔로우업하여 구조 및 방식을 이해하고 이후 구현하기 위해 테스트 코드를 작성 후 테스트를 통해 좀 더 직접적으로 이해하였습니다.

2. 나는 어떤 방식으로 모델을 개선했는가?

- 여러 모델 클래스 분류로 하나의 파이프라인 형태에서 하나의 클래스로 원하는 모델을 불러서 이를 G-eval 평가지표로 파악 가능하게 하여 모델 선정을 가능하게 하였습니다.
- Agent 의 테이블 + 텍스트 입력으로 원하는 형태의 그래프, 표 이미지를 생성할때 Seaborn , Matplotlib 의 혼용 및 추가 프롬프팅으로 좀 더 논문에서 사용하는 형태의 이미지를 생성하도록 개선하였습니다.
- Agent 구조에서 초기 단계에 synchronous 한 구조를 가지고 있어 Agent 의 CoT 및 multi agent 된 프로젝트 구조상 output 의 interval 이 매우 길어지게 되어 이들을 각자의 async 하게 처리하고 해당 작업을 마친 후 FastApi 에 넣어 서비스를 구현함으로서 좀 더 빠른 output 이 나오도록 개선하였습니다.
- 서비스에서 각 이미지들을 저장한 POST 방식의 url 을 직접 접근하지 않고 이들을 심볼릭 링크하여 연결 후 사용자들이 생성한 이미지, pdf 들을 따로 다운받을 수 있는 형태로 제공하여 서비스적인 편의성을 좀 더 개선하였습니다다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

- 사용자의 질의에 대하여 참고 문서의 정확한 이름과 참고 문서의 페이지를 같이 참조하는 형태와 텍스트와 연관지어 참조하기 좋은 구조인 그래프, 표 형태의 이미지를 제시하여 서비스가 사용자 뿐만 아니라 전문가들도 사용할 수 있는 구성으로 제시를 하였으며 해당 과정에서 Parsing 의 형태, Retrieval 의 구성이 매우 유기적으로 연결되어야 한다는 점과 사용자의 입장에서 어떠한 점이 매력있는 서비스 형태인지 고민해보게 되었습니다.

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 대형 모델 서빙을 위하여 분산처리를 시도하였고 기존의 8b 모델이 아니라 32B 모델까지 서빙하는 테스트를 진행하였는데 Qwen, Exaone 등의 모델을 32B 에서 서빙해보았습니다. 그리고 해당 모델의 결과는 의외로 한국어 주식 관련 도메인에서의 G-eval 은 Exaone 7.8B 가 Unsolth, 기존 Qwen 32B 모델보다 더 좋았으며 이는 GPT 4o 의 성능과도 유사했습니다.
- 또한 Retrieve 과 유기적으로 연결하기 위해 데이터의 구조를 좀더 개선하여 이를 연결했을 경우 summary 에서 context 를 요약한 내용을 기반으로 서치 후 이를 context 의 결과로 유도하니 좀 더 빠른 응답 속도를 보였습니다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 폐쇄된 ip 서버 환경에서는 결국 초기 각기 서버에 접속한 환경으로 세팅하여 터널링을 구성한 후 포트포워딩으로 서버를 서로 연결하였는데 해당 방법으로 분산 처리할 경우 맨 처음에는 모델을 다운받는 부분에서 타임아웃이 발생하여 각 서버별 사전 모델을 다운받은후 진행하였는데 해당 부분에서 서버의 크기가 100G 의 제약조건이 발생하여 사전 가중치 모델이 양자화된 형태가 아니면 32B 이상의 대형모델은 사용에 제약이 발생하였으며 또한 분산 추론시 터널링의 문제로 응답이 도중에 끊길 수도 있는 불안정성이 존재하여 제대로 활용하지 못한 문제가 있었습니다.
- 또한 멀티턴 방식의 DB를 구축하지 못해 하나의 대화구성에서 장기적인 질의에 대해 연관성있는 대답을 지원하지 못한 것이 서비스적으로 조금 아쉬웠던 것 같습니다.

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- Open ip 서버 환경에서 분산처리로 대형 모델 서빙 방식 시도를 함으로써 좋은 대답을 빠르게 낼 수 있는 환경을 구성하고함
- 단순한 LLM 모델 뿐만 아니라 멀티 모달 모델, 강화학습 모델, Finetuning 모델을 같이 시도하면서 이를 멀티 턴 형태로 구현함으로서 다양한 형태의 대답을 처리가능하도록 구성하고자 함
- Unlearning 을 사용하여 기존의 Finetuning 된 대형 모델에서 내가 원하는 방향으로 모델을 재조정하여 학습 과정의 비용을 최적화하는 방향으로 진행해보고자 함

정유진 (T7435)

1. 나는 내 학습목표 달성을 위해 무엇을 어떻게 했는가?

금융 도메인에서 활용할 수 있는 RAG 기반 LLM 챗봇을 개발하였습니다. 이를 개발하기 위해 Multi-Agent 방식을 적용하여 사용자 친화적인 서비스 개발을 목표로 하였습니다.

- PDF Parsing : 금융/증권 보고서 PDF 데이터를 효율적으로 저희 데이터베이스에 적재하기 위해 텍스트 뿐만 아니라 표와 이미지도 잘 추출할 수 있는 Parsing 방식을 조사하였습니다. 조사 중 발견한 pdfplumber, py-pdf2 등 다양한 파이썬 라이브러리를 테스트하였습니다.
- 벡터 Database : 검색의 정확도와 속도를 높이기 위해서 벡터 DB와 그래프 DB 중 고민을 하였고, 우리가 가진 데이터의 특성과 서비스의 목표에 맞춰 벡터 DB 방식을 선택하였습니다. 그 중 로컬에서도 적용하기 편한 ChromaDB 라이브러리를 활용하여 DB를 구축하였습니다.
- Embedding : 데이터와 사용자의 쿼리를 벡터화하기 위해 다양한 임베딩 모델을 테스트해보았습니다. 저희 팀의 가설로는 금융권 데이터이다 보니 금융 특화 임베딩 모델을 사용하는 것이 검색 정확도가 높을 것이라고 예상했지만, 예상 외로 NAVER CLOUD 임베딩 모델이 더 높은 정확도를 보였습니다.
- AI Agent 구축 : 다양한 서비스를 제공하기 위해 Multi-agent 방식을 채택했고, CrewAI라는 툴을 활용하여 기반을 구축하였습니다.
- Frontend 개발 : React와 Tailwindcss를 활용하여 서비스의 웹 화면을 개발하였습니다.

2. 나는 어떤 방식으로 모델을 개선했는가?

- Embedding 모델 평가 : Embedding 모델 중 금융 특화 모델인 KLUE와 kf-deberta-base 모델, 그리고 LLM 모델, 네이버 클라우드의 임베딩 모델을 테스트하여 검색 시 가장 높은 정확도를 보이는 모델을 선택하여 개발을 진행했습니다.

3. 내가 한 행동의 결과로 어떤 지점을 달성하고, 어떤 깨달음을 얻었는가?

- PDF Parsing을 수행할 때, pdfplumber라는 라이브러리를 이용하였습니다. 높은 인식과 추출율을 기대하였지만, 표의 형식이 정확하지 않을 경우 제대로 파싱하지 못한다는 단점이 있었습니다.
- CrewAI를 활용하여 Agent를 구현했는데, response 속도가 빠르지 않았습니다. 이는 CoT가 여러 번 수행되고, 동기적으로 수행할 시에 느려진다는 점을 깨달았습니다. 또한 불필요한 부분까지 Agent를 활용하기 때문에 처음 구상을 좀 더 꼼꼼하게 하고, 다른 툴의 활용은 어떻게 되는지 알아봐야겠다는 생각을 하였습니다.

4. 전과 비교하여 새롭게 시도한 변화는 무엇이고, 어떤 효과가 있었는가?

- 협업을 위해 GitHub의 Discussion, Issue, PR 기능을 활발히 활용하였고, 이러한 시도들은 전 프로젝트보다 더 높은 협업률과 프로젝트 진행률을 보였습니다.
- 저번 프로젝트에서 Agent를 활용해보고자 하는 의지를 가지고 있었지만 실제 적용은 하지

못했는데, 이번 프로젝트에서는 Agent를 메인으로 개발을 수행하여 해당 내용을 제대로 공부하고 체화시키는 계기가 되었습니다.

5. 마주한 한계는 무엇이며, 아쉬웠던 점은 무엇인가?

- 여러 가지 역할 중 프론트엔드 개발 부분에서 아쉬움이 남습니다. JavaScript와 React를 이번 프로젝트에서 처음 사용하는 것이다 보니, LLM 모델 없이 혼자서 개발하는 것이 힘들었습니다. 최소한의 개발과 디자인은 완수해냈지만, 지식 부족으로 더 퀄리티 높은 웹 페이지를 개발하지 못한 것에 아쉬움이 남습니다.

6. 한계/교훈을 바탕으로 다음 프로젝트에서 시도해볼 것은 무엇인가?

- 프론트엔드와 백엔드에 대한 섬세한 개발과 배포까지 진행하여 실제 사용자들이 저희의 서비스를 사용해보면서 피드백을 받고 싶습니다.
- 또한 Autogen과 Langraph 등을 활용하여 사용자들이 주는 데이터를 저희 서비스 DB에 적재될 수 있는 자동화 시스템 개발도 해보고 싶습니다.

Reference

1. https://docs.trychroma.com/docs/overview/introduction
2. https://python.langchain.com/docs/get_started/introduction
3. https://fastapi.tiangolo.com/
4. https://react.dev/
5. https://tailwindcss.com/
6. https://platform.openai.com/docs/models/gpt-4o-mini
7. https://exaone.lge.com/exaone-api
8. https://qwen.baidu.com/doc/index
9. https://docs.crewai.com/concepts/tools
10. https://github.com/upskyy/kf-deberta-multitask
11. https://www.ncloud.com/
12. https://console.upstage.ai/docs/capabilities/document-parse