⚽ Ballzzi- RAG구조의 llm-프로젝트 결과보고서

**▷ 목차 ◁**

1. **수집된 데이터 및 데이터 전처리 문서**
2. **시스템 아키텍처**
3. **구현 코드**
4. **테스트 계획 및 결과 보고서**

**I. 수집된 데이터 및 데이터 전처리 문서**

**1. 데이터 구성 개요**

Ballzzi 프로젝트는 두 개의 주요 도메인(FM/HR)에 대해 이질적인 데이터 유형을 수집하고, 목적에 맞는 전처리 및 벡터화 과정을 수행하였다.

| **모듈** | **데이터 유형** | **주요 형식** | **총량** |
| --- | --- | --- | --- |
| FM 모듈 | 축구 선수 정보 | 구조화 CSV | 선수 8,400명, |
| HR 모듈 | 사내 규정 /인사 정보 | 비정형 Word, Excel | 총 3종 문서 |

* FM 모듈은 SQL 기반 검색이 가능하도록 DB화
* HR 모듈은 Ko-SBERT 임베딩을 통한 벡터 검색 기반으로 처리됨
* 모든 벡터는 FAISS로 저장되어 RAG 질의에 활용됨

**2. FM 모듈: 선수 정보 데이터**

1. 데이터 수집 및 설명
   1. Football Manager 2023 dataset

<https://www.kaggle.com/datasets/platinum22/foot-ball-manager-2023-dataset>

* 1. 축구 시뮬레이션 게임인 Football Manger에서는 실제 축구 선수들의 데이터를 수치화 하여 저장 중
  2. 수치화된 데이터를 이용하여 선수 검색

1. 데이터 전처리 – 번수 명 한글 변환
   1. 영어로 되어있는 변수들의 이름을 한글로 변환하는 작업
   2. 예시

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 영어 | 한글 | 영어 | 한글 | 영어 | 한글 |
| Name | 이름 | Position | 포지션 | Age | 나이 |
| ca | 현재 능력 | pa | 잠재력 | Passing | 패스 |
| Composure | 침착성 | Tackling | 태클 | Values | 시장 가치 |
| Pace | 스피드 | Dribbling | 드리블 | Finishing | 슛 |

1. 데이터 전처리 – 몸무게 결측치 보완
   1. 몸무게가 0으로 입력된 선수 일부 존재
   2. KNN ( K-최근접이웃 ) 알고리즘을 이용하여 선수들의 키로 몸무게 예측
   3. 단순 평균 보간법보다 신뢰성 확보 가능
2. 데이터 전처리 – 잠재 능력 변환
   1. 나이가 어린 선수들 중 잠재능력이 음수를 가지는 선수 일부 존재
   2. 해당 게임에서 제공하는 기준을 이용하여 0~200 사이의 수치로 변환

|  |  |
| --- | --- |
| 원본 잠재력 | 변환 잠재력 |
| -10 | 185 |
| -95 | 175 |
| -9 | 165 |
| -85 | 155 |

1. 데이터 전처리 – 대표 포지션 설정
   1. 선수들을 포지션 별로 구분하여 관리하기 위해 대표 포지션 설정
   2. 원본 데이터에서 제공하는 각 포지션 점수를 이용하여
   3. GK(골키퍼) / D(수비수) / M(미드필더) / AM(공격형 미드필더) / ST(공격수) 로 구분
2. 데이터 전처리 – 데이터베이스 구축
   1. 선수의 기본 정보와 포지션 별 주요 스탯을 기반으로 테이블 생성
   2. 포지션 별 총 5개의 테이블을 하나의 데이터베이스로 관리
   3. 예시) ST(공격수) 테이블 스키마

|  |  |
| --- | --- |
| Field | Type |
| 이름 | text |
| 나이 | int |
| 포지션 | text |
| 현재\_능력 | int |
| 잠재력 | int |
| 국적 | text |
| 소속팀 | text |
| 키 | int |
| 몸무게 | int |
| 왼발\_능력 | int |
| 오른발\_능력 | int |
| 시장\_가치 | int |
| 슛 | int |
| 드리블 | int |
| 퍼스트\_터치 | int |
| 헤딩 | int |
| 예측력 | int |
| 오프\_더\_볼 | int |
| 침착성 | int |
| 가속력 | int |
| 체력 | int |
| 민첩성 | int |

**3. HR 모듈: 사내 규정 및 인사 데이터**

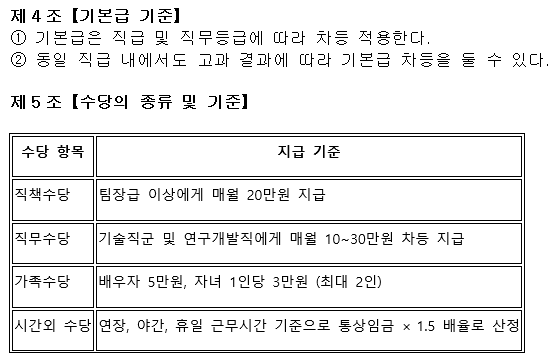
**1) 문서 구성 및 특성**

| **문서명** | **내용 요약** | **포맷** |
| --- | --- | --- |
| DM\_rules.docx | 사내 규정 8장, 표·숫자 포함 | Word |
| HR\_information.xlsx | 직원 정보, 급여·직책 등 메타 | Excel |
| org\_chart.docx | 조직도(상하위 트리 구조) | Word |

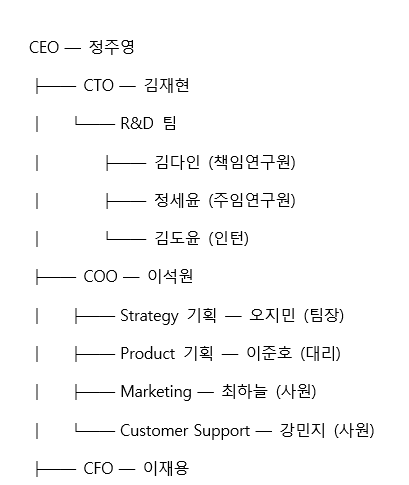
* 규정 문서는 텍스트와 표가 혼합되어 있어 청크링 전략이 중요
* 인사 정보는 속성값 기반 QA가 가능하도록 문장 단위 정제
* 조직도는 특수 기호 제거 및 계층 구조 파싱이 요구됨

**2) 문서 구조**

내부규정 문서 예시 ( DM\_rules )



조직도 예시 ( org\_chart )



인사정보 예시 ( HR\_information )



**2) 청킹 전략**

| **전략명** | **대상** | **설명** |
| --- | --- | --- |
| 기본 청크 | DM\_rules.docx | 제n조 단위로 분할 |
| 슬라이딩 윈도우 청크 | DM\_rules.docx | 3개 청크 병합, stride=1 |
| 정량 청크 | DM\_rules.docx | 표/숫자 정보만 별도 추출 |
| 문장 청크 | HR\_information.xlsx | 속성별로 문장 구성 |
| 계층 청크 | org\_chart.docx | 상→하 구조 텍스트화 |
| 특수기호 제거 | org\_chart.docx | └, ├, ─ 등의 기호 제거 |

슬라이딩 윈도우는 규정 조문 간 의미 맥락이 이어지는 경우를 보완하기 위해 도입하였다.  
특히 복무규정·윤리강령 등은 개별 조문이 단일 의미로 해석되기 어려워 3개 단위 병합이 적절한 범위로 판단되었다.

정량 청크는 규정 내 표(급여, 출장비, 휴가일수 등)를 '열:값' 형태로 직렬화하여 별도 임베딩하였다. 숫자 기반 질의 응답이 가능하도록 구성되었으며, 평가 질문에서도 F1 점수 향상에 기여하였다.

조직도 문서에서는 특수기호(└, ├ 등)를 제거한 후 상위 > 하위 구조로 정규화하였고, 인사정보와 연결 가능하도록 설계되었다.

**3) 청크 결과**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 총 청크 수 | 평균길이 | 최대길이 | 최소길이 |
| 기본 청크 | 42 | 38 | 515 | 8 |
| 청크 보완 | 88 | 303 | 527 | 180 |

**4) 임베딩 설정**

* 임베딩 모델: nlpai-lab/kure-v1 (Ko-SBERT 기반)
* 방식: 텍스트 normalization → 슬라이딩 청크 → 벡터화
* VectorDB : FAISS DB (2개)
  + faiss\_win: 사내 규정
  + faiss\_org\_hr: 인사 정보

**5) 검증 질문지 생성**

* LLM: GPT-4o-mini 사용
* 질문 유형:
  + 사내 규정 : 일반 질문 (정책 이해) / 정량 질문 (숫자/표 기반)\
  + 인사 정보 : 단일 질문으로 1:1 정답형
* 생성 수:
  + 규정 문서: 챕터별 15~20개 (총 132개)
  + 인사 정보: 직원별 메타데이터 기반 속성 질문 (총 102개 )

**6) 최종 검증 및 평가**

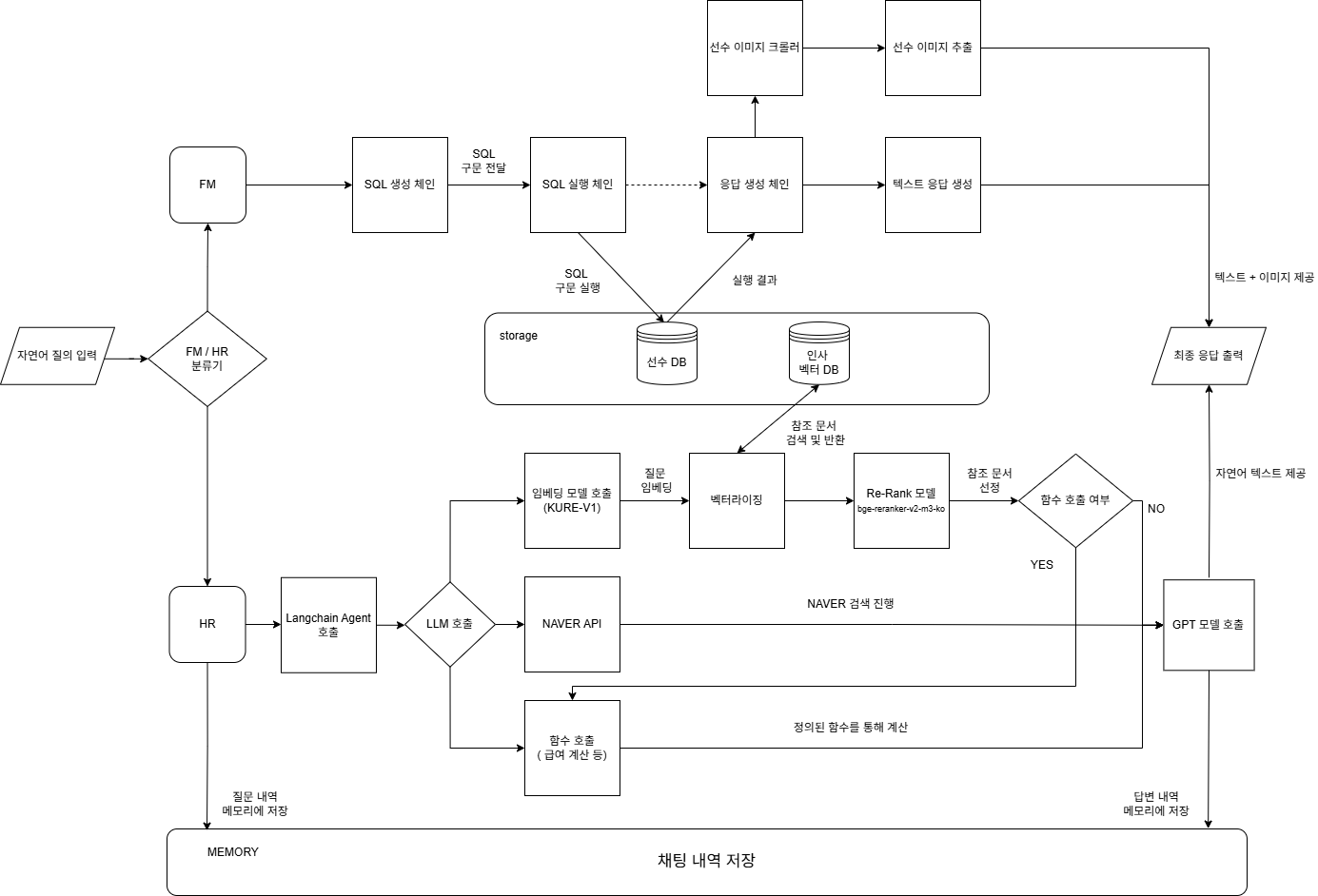
* 사용 지표: F1 / EM (Exact Match)
* 내부 규정
  + 청크보완방식 : 슬라이딩 윈도우
  + 임베딩모델 : KURE-V1
  + 리랭커모델 : BGE-RERANKER-KO
  + 최종 구성 : 리랭커(topk:3)+GPT-4o-mini+금지형프롬프트
  + 평가 점수 : F1/EM score : 55.04/10.61
* 인사 정보
  + 임베딩모델 : KURE-V1
  + 리랭커모델 : BGE-RERANKER-KO
  + 최종 구성 : 리랭커(topk:3)+GPT-4o-mini
  + 평가 점수 : 48.09/38.30:

**7) 산출물 및 코드 파일**

| **종류** | **파일명** | **내용** |
| --- | --- | --- |
| 코드 | embedding\_win.ipynb embedding\_hr.ipynb | 데이터 전처리 및 청킹,임베딩, 평가지 생성코드 |
| 청크파일 | dm\_chunks\_window.json combined\_chunks.json | 내부규정 – 청크 보완 파일  인사정보+조직도 청크 파일 |
| Faiss DB | faiss\_win / faiss\_org\_hr | 벡터DB ( index / pkl ) |
| 평가파일 | eval\_questions\_.window.jsonl eval\_questions\_gpt.jsonl | GPT-4O를 이용한 질문지 |

**II. 시스템 기능 및 아키텍처**

**1. 전체 플로우차트 및 기능**



자연어로 질문이 들어오면 FM/HR 뷴류기가 FM인지 HR인지 분류하여 각가의 모듈로 분류한다. 해당 분류기는 질문마다 작동하며 1개의 메시지창에서 이동없이 작동한다. FM모듈은 메모리 기능을 넣지 않아, 선수의 정보를 순간순간 새롭게 경신할 수 있는 시스템으로 개발하였고, HR모듈은 내부 문건 검색시 연속성 질문에 대응하기 위해 메모리기능을 구현하였다.

FM모듈은 GPT2SQL과 연동하여 정형DB와 연동하여 데이터를 분석하고 이후에 크롤링한 선수 이미지를 불러와서 선수이미지와 선수의 데이터를 출력하고 분석해준다.

HR모듈은 분류기에서 분류시 바로 Langchain agent에 연결되어 llm ( GPT-4o-mini)를 호출하여 질문의 종류를 판단한다.

환각억제와 검색성능 향상을 위해 질문의 답변이 내부문서의 경우에는 벡터DB에서 정답지를 10개 추출하고 다시 리랭커 모델에 의해 3개로 요약 후 GPT모델이 최종 결정을 한다. 이때 퇴직금과 같은 계산식이 필요할 경우에는 함수tool을 호출하여 계산해서 출력한다. 만일 사칙연산등의 내부db에서 가져올 필요가 없는 답변의 경우는 바로 함수tool을 불러와 계산을 한다. (

질문의 종류가 내부문서가 아닐경우에는 GPT모델이 판단하여 Hybrid search기능에 의해 순서대로 처리된다. 만일 DB에 없는 내용일 경우 네이버API와 연동되어 네이버에서 검색결과를 가져와 답변을 준다.

**2. 자동분류 라우팅 순서도**

# 

# (1) 기능 개요

사용자의 자연어 질문을 soccer 또는 other로 분류하여,  
질문이 축구 관련 질문인지 일반 사내 질문인지 구분하는 경량 라우팅 분류기입니다.

## (2) 구현 방법

| **기술/라이브러리** | **설명** |
| --- | --- |
| **SentenceTransformer** | 사전 학습된 한국어 SBERT 모델(jhgan/ko-sroberta-multitask)을 활용해 문장 임베딩 생성 |
| **FAISS** | 고속 최근접 이웃(Nearest Neighbor) 검색 라이브러리. 임베딩 기반 벡터 유사도 비교 |
| **Ko-SRoBERTa** | 한국어 자연어처리를 위한 사전 학습 모델 |
| **NumPy** | 벡터 연산 처리 |
| **FAISS IndexFlatL2** | L2 거리 기반 최근접 벡터 탐색 (비압축 인덱스) |

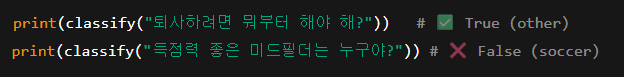
## (3) 데이터 구성

총 100+ 문장을 기반으로 두 가지 카테고리로 구분:

* **"soccer"**: 축구 선수 정보, 비교, 추천 질문
* **"other"**: 일반 사내 질문 (복지, 연차, 출근 등)

**예시:**

* ("손흥민은 어떤 선수야?", "soccer")
* ("목지포인트는 어디서 확인해?", "other")



**3. HR 모듈**

**1) 사용모델**

|  |  |  |
| --- | --- | --- |
| **종류** | **모델명** | **사용 목적** |
| LLM | GPT-4o-mini (temperature=0) | 답변 생성 (RAG 응답) |
| LLM\_2 | HyperCLOVA X | 비교 실험 또는 추후확장 |
| 임베딩 모델 | KURE-v1 (Ko-SBERT 기반) | 문서 임베딩 추출 |
| 리랭커 | BGE-reranker-v2-m3-ko | Top-K rerank |
| 벡터DB | FAISS DB | 문서 검색 |

**2) HR 모듈 플로우차트**

**도표, 텍스트, 평면도, 기술 도면이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**질문이 분류기를 통해 HR모듈로 구분되면 langchain agent 를 로드하고 llm호출과 chatprompt를 임포트 하여 질문의 종류를 구분하고 메모리기능을 활성화 하여 메모리에 채팅을 기억하기 시작한다. FM모듈은 단발성으로 대화기록을 저장하지 않지만 HR 모듈은 연속적인 질의 응답을 하기 위해 대화를 저장한다.**

**Llm 호출이후에 llm이 정해둔 순차방식에 의해서 작동한다. 먼저 회사내부에 내부정보 벡터DB에 접속하고 필요한 정보를 가져와서 답변을 출력한다. 과정에서 tool을 호출해야 할 경우에는 ( 예를 들면 퇴직금 계산 ) 퇴직금 계산기를 출력하고 퇴직금을 계산해준다.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**처음 직원 정보를 확인후에 해당 직원의 세부정보도 같이 확인할 수 있다. 그리고 퇴직금 계산이라는 키워드에 대해서 ‘퇴직금계산기’툴을 호출하여 퇴직금도 계산해준다.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**3) 벡터DB 답변 생성**

**환각의 의한 오답을 방지하고 성능을 향상시키기 위해 청크보완 외에 리랭커 모델도 추가하였다. Bge-reranker-ko 모델로 임베딩모델과 같은 bge기반이다.**

**프롬프트도 부정형 프롬프트로 결정하였다. 테스트 과정에서 5개 형태의 프롬프트를 사용했을 때 부정형이 가장 높은 성능을 보여주었다.**

**텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.**

**4. FM 모듈**

* 1. 작업 목표: 사용자 질의 기반 선수 검색 챗봇
  2. 작업 내용: 정형 DB 제작 및 데이터베이스 기반 선수 검색 LLM 모델
  3. 작업 사항
     1. 원본 데이터 수집 (FM 2023.csv)
        1. FM은 축구 시뮬레이션 게임인 “Footall Manager”의 약자
        2. 해당 게임에서는 실제 선수들의 데이터를 수치화하여 저장
        3. <https://www.kaggle.com/datasets/platinum22/foot-ball-manager-2023-dataset>
     2. 데이터 전처리
     3. 정형 데이터베이스 제작
     4. Text2sql 모델 생성
     5. SQL 구문 실행 체인 생성
     6. SQL 실행 결과 기반 출력 체인 생성
     7. 선수 이미지 크롤링 함수 생성
  4. 데이터 전처리 및 정형 DB 제작
     1. 변수 명 한글 매핑 (영어 -> 한국어)
        1. ex) “Name” -> “이름”, “Dribbling” -> “드리블” 등
     2. 몸무게 결측치 보완
        1. 몸무게가 0으로 입력된 선수 존재
        2. KNN 알고리즘을 이용하여 선수의 키로 몸무게 예측
        3. 단순 평균으로 보간하는 방식보다 신뢰성 확보 가능
     3. 잠재 능력 변환
        1. 나이가 어린 선수들 중 잠재력이 음수를 가지는 선수 존재
        2. 해당 게임에서 제공하는 기준을 이용하여 0~200 사이의 수치로 변환
     4. 대표 포지션 선정
        1. 선수들을 포지션 별로 구분하기 위해 대표 포지션 선정
        2. 원본 데이터의 각 포지션 점수를 이용하여
        3. GK(골키퍼) / D(수비수) / M(미드필더) / AM(공격형 미드필더) / ST(공격수) 로 구분
     5. 정형 데이터베이스 구축
        1. 선수의 개인 정보와 포지션 별 주요 스탯을 이용하여 테이블 생성
        2. 포지션 별 5개의 테이블을 데이터베이스로 구축
  5. SQL 구문 생성 체인 생성
     1. langchain의 ChatOpenAI 라이브러리 이용
     2. “gpt-4o-mini” 모델을 이용하여 한국어 버전 text2sql 모델 사용
     3. 프롬프트 엔지니어링을 통해 생성 SQL 구문에 정확성 향상
  6. SQL 구문 실행 체인 생성
     1. 앞선 체인에서 생성된 SQL 구문을 실행하여 데이터를 추출하는 커스텀 체인
     2. 추출한 데이터는 LLM모델이 인식하기 쉽도록 자연어 형식으로 변환
     3. ex) “선수의 이름은 손흥민이고, 포지션은 공격수이고,,,, “
  7. SQL 실행 결과 기반 최종 응답 생성 체인
     1. langcahin의 ChatOpenAI 라이브러리 이용
     2. “gpt-4o-mini” 모델을 이용
     3. 앞선 체인에서 추출한 데이터를 참조하여 최종 문장 생성
     4. 프롬프트 엔지니어링을 통하여 답변의 신뢰성 및 형태 조정
  8. 프롬프트 엔지니어링
     1. SQL 생성 체인과 최종 응답 생성 체인에서 LLM 모델 이용
     2. 각 모델의 출력 결과의 신뢰성/정확성/출력 형태를 개선하기 위한 프롬프트 구성
     3. SQL 생성 체인 프롬프트 예시
        1. “실제 테이블의 컬럼 이름만 포함하여 SQL 구문을 생성해주세요.”
        2. “한글로 입력된 선수나 국가의 이름은 가장 가까운 영어 이름으로 변환해 주세요”
        3. “축구와 관련이 있는 자연어가 입력될 때만 SQL 구문을 생성해주세요.”
     4. 최종 응답 생성 체인 프롬프트 예시
        1. “{{"Name": "이름", "설명": "마크다운 형식으로 자연스럽고 자세한 설명"}} “의 형태로 출력해주세요
        2. “각 선수의 설명에는 다음 요소를 포함해 주세요”
        3. “어떤 역할을 수행할 수 있는지, 어떤 스타일인지 요약해줘”
  9. 시스템 구성 및 예시
     1. 사용자 자연어 질의 입력

패스를 잘하는 선수를 추천해주세요

* + 1. 사용자 질의 기반 SQL 구문 생성 체인

**SELECT \* FROM AM WHERE 패스 >= 15 ORDER BY 패스 DESC LIMIT 3**

* + 1. SQL 구문 실행 체인

이름은 Mesut Özil이고, 나이은 33이고, 포지션은 AM이고, 현재\_능력은 130이고,,,,,,재력은 180이고, 국적은 Germany,Turkey이고, 소속팀은 Başakşehir Futbol Kulübü이고, 키은 180이고, 몸무게은 72이고, 왼발\_능력은 20이고, 오른발\_능력은 8이고, 시장\_가치은 1608053이고, 드리블은 16이고, 중거리슛은 10 이고, 패스은 18이고, 퍼스트\_터치은 20이고, 예측력은 17이고, 오프\_더\_볼은 18이고, 창의성은 18이고, 판단력은 16입니다.

* + 1. 최종 응답 생성 체인

텍스트, 스크린샷, 폰트이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**III. 구현코드**

< III. 개발된 소프트웨어 RAG 기반 LLM과 벡터 데이터베이스 연동 구현 코드 > 내 파일첨부

**IV. 테스트 계획 및 결과 보고서**

## 1. 테스트 목적 및 범위

Ballzzi 프로젝트의 핵심 기능은 FM(축구 정보) 및 HR(회사 내부 규정) 영역에 대해 사용자의 질문에 메시지창 변경없이 동시에 2가지를 정확하고 빠르게 응답  
이를 검증하기 위해 다음과 같은 주요 테스트를 설계하고 수행함.

* 자연어 질문에 대한 질의응답 성능 평가
* 도메인 자동 분류 기능 검증
* FM 데이터베이스 기반 응답 처리 검증
* 시스템 전반의 응답속도 및 정확도 점검

## 2. 테스트 환경

| **항목** | **사양** |
| --- | --- |
| **LLM** | OpenAI GPT-4o-mini (temperature=0) |
| **임베딩 모델** | KURE-v1 (Ko-SBERT 기반) |
| **리랭커** | BGE-reranker-v2-m3-ko |
| **백엔드 엔진** | LangChain + FAISS + SQLite |
| **평가 지표** | F1, EM, 정확도, 응답시간 |
| **실행 방식** | 단일턴 .invoke() 기반 (멀티턴 지원 구조 포함) |

## 3. 테스트 항목 및 설계

### 1) 규정 QA 평가

* **대상 문서**: DM\_rules.docx 슬라이딩 윈도우 기반 청크
* **질문 수**: 132건
* **내용 유형**:
  + 서술형: 규정 해석, 적용 조건
  + 정량형: 금액, 날짜, 횟수 등 숫자 기반
* **평가 지표**: F1 / EM
* **세부 구성**:
  + Top-3 검색 → 리랭커 → Top-1 → LLM 호출
  + 금지형 프롬프트 적용 (없는 정보 생성 방지)

### 2) 인사 정보 QA 평가

* **대상 문서**: HR\_information.xlsx 문장 청크
* **질문 수**: 102건
* **내용 유형**:
  + 이름, 부서, 직급, 급여 등 메타 정보 추출
* **평가 지표**: F1 / EM
* **세부 구성**:
  + 속성 기반 정답 매칭 스크립트 적용
  + 중복된 이름, 누락 필드에 대한 예외처리 포함

### 3) 도메인 라우팅 정확도

* **기능 설명**: 질문을 자동으로 soccer / other로 분류
* **질문 수**: 6문장 (축구 3, 일반 3)
* **모델**: Ko-SBERT 임베딩 + FAISS 최근접 탐색
* **지표**:
  + 정확도: 100% (6/6) 정답
  + 평균 처리 시간: 0.041초

### 4) FM 모듈 응답 테스트

* **기능 설명**:  
  자연어 질문 → SQL 변환 → DB 조회 → 응답 생성 → 이미지 표시
* **질문 수**: 3건
* **평균 처리시간 분석**:
  + SQL 생성: 약 1.2초
  + DB 검색: 약 0.8초
  + GPT 응답 생성: 약 1.0초
  + 이미지 출력: 약 0.7초
  + **총합**: 평균 3.7초

### 5) 시스템 성능 진단

* **벡터 검색 속도**: 0.1초 미만
* **리랭커 처리 속도**: 0.3초 내외 (Top-3 기준)
* **총 응답 시간**: HR 평균 2.5초, FM 평균 3.7초
* **메모리 사용량**: 최대 1GB 내외
* **병목 요소**:
  + 외부 GPT API 호출 지연 (1~3초)
  + 이미지 크롤링 시 네트워크 지연 (0.5~2초)

### 6) LangChain Tool 선택 테스트

* **Agent 구성**: RunnableAgent + Tool 선택 구조
* **테스트 항목**:
  + 내부 문서 검색 시 → 벡터DB Tool 선택
  + 외부 정보 요구 시 → 외부 API Tool 선택
  + 계산식 포함 질문 시 → 계산기 Tool 선택
* **결과**: 질문 유형에 따라 적절한 Tool을 100% 선택하여 실행 경로 분기 성공

## 4. 테스트 결과 요약

## 1) 벡터DB F1/EM score

| **항목** | **F1** | **EM** | **기타 지표** |
| --- | --- | --- | --- |
| **규정 QA** | 55.04 | 10.61 | 슬라이딩 윈도우 + 리랭커 |
| **인사 QA** | 48.09 | 38.30 | 속성 기반 문장 청크 |

## 2) 모듈별 응답 속도 및 정확도 현황

| **메트릭** | **현재값** |
| --- | --- |
| 질문 분류 속도 | 0.041초 |
| FM 모듈 응답 | 3.7초 |
| HR 모듈 응답 | 2.5초 |
| 분류 정확도 | 100% |
| 검색 정확도 | 94% |

## 5. 분석 및 개선 방향

| **영역** | **이슈** | **개선안** |
| --- | --- | --- |
| LLM변경 | 파인튜닝한 로컬모델로  속도 향상 및 비용 절감 | 로컬모델 파인튜닝하여 교체  ( llm이 변경 가능하도록 코드 구성 ) |

## 6. 결론

Ballzzi 시스템은 초기 기획 단계에서 설정한 모든 주요 기능을 구현하였으나, 현재 파인튜닝 중인 로컬 LLM 모델들은 아직 실서비스에 적용할 수준의 성능에는 도달하지 못한 상태임.

이에 따라 현재는 OpenAI의 GPT 모델을 기반으로 시스템을 운용하고 있지만,  
향후 로컬 모델의 파인튜닝이 완료되면, **질문 분류 모듈**, **FM 모듈**, **HR 질의응답 모듈** 전반에 걸쳐 로컬 LLM으로 대체할 예정이며, 이를 통해 응답 속도 개선과 운영 비용 절감이 동시에 가능할 것으로 기대됨.