



Football Agora

[우리만의 풋살스토리를 직접 기록하고 공유할 수 있는 플랫폼]

1. 개요

- 프로젝트 소개
- 서비스 기획 배경
- 팀원 소개

3. 개발 상세

- 서비스 기능 상세
- 웹서비스 개발
 - i) 유저기능
 - ii) 서비스 기능

5. 결과

- 결과물
- 추후 과제

2. 기획 및 추진 계획

- WBS
- 기획자료
- DB 아키텍처

4. 주요 기능

- 1) 영상분석
- 2) llm & Chatbot
- 3) 모델 서빙 및배포

6. 부록

- 개발 환경 및 라이브러리
- 참고문헌/data 출처

A soccer ball with blue and white patterns is positioned on a green grass field, resting on a white boundary line. In the background, a soccer goal is visible under a clear sky. A semi-transparent circle containing the number '1' is centered over the image.

1

개요

서비스 기획 배경



김하니(24세, 대학생):
풋살에 이제 관심을 가지기 시작했는데...
규칙이나 잘하는 법을 모르겠어

이민호(34세, 풋살동호인):
프로 스카우터처럼 나도 내 경기를 분석하고
피드백해서 실력을 키우고 싶어!!



서비스 기획 배경

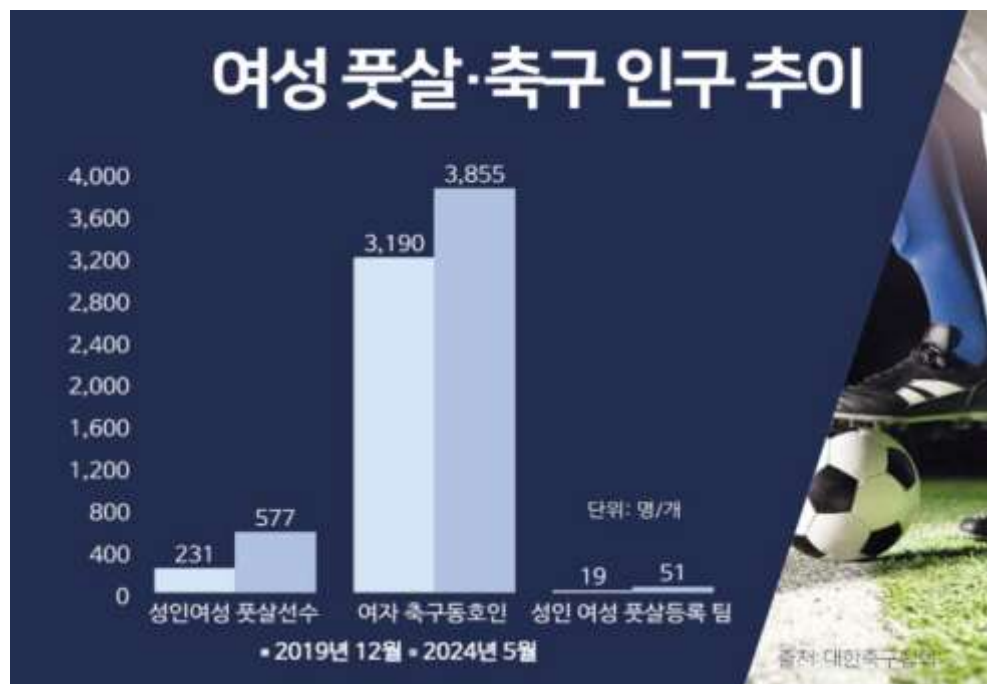
1 기획배경



풋살 수요 증가



너도 나도 '골때녀'... 국내에 부는 여성 풋살 열풍



'축구의 성벽 허물다' 여성 풋살팀 5년새 3배 급증



서비스 기획 배경

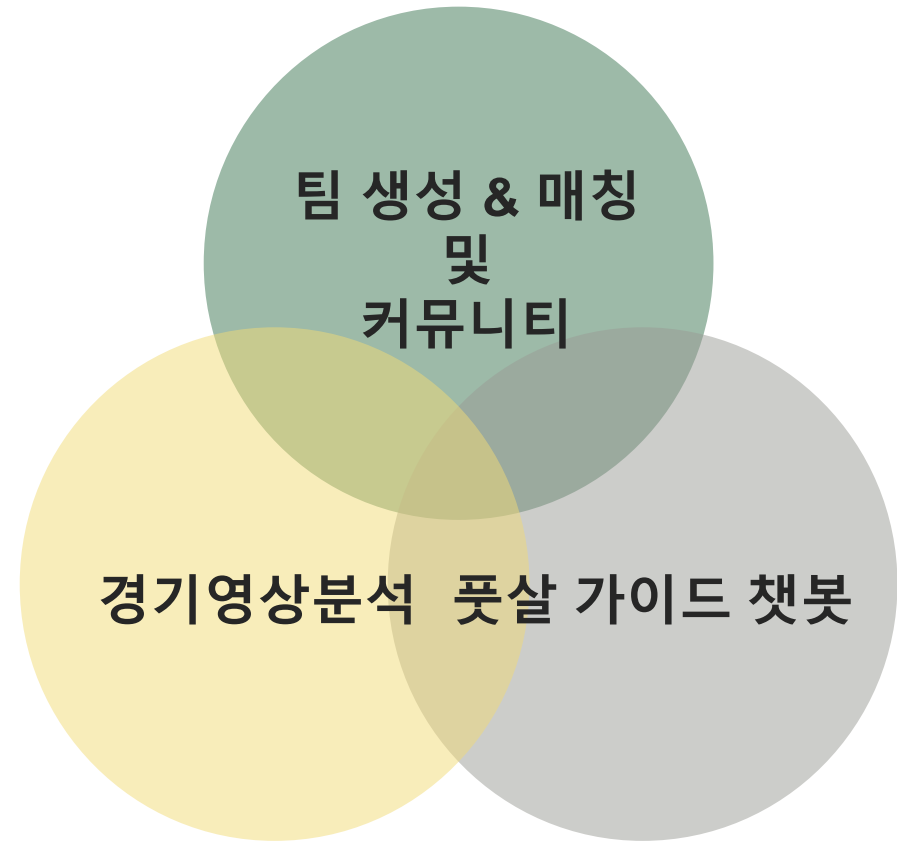
2 기획배경 및 기대효과



이미지 딥러닝 기술을 활용해서
팀 경기 영상분석 서비스를 제공함에 따라
풋살을 하는 일반인 팀에서도 각 포지션에 따른 경기 분석을 통해
팀 경기를 피드백하고 팀 커뮤니티를 보다 활성화 시킬 수 있다.



거대언어 모델 기반 생성형 AI 챗봇을 제공하여
풋살을 처음 접하는 사람들도 궁금한 점을
쉽게 물어보고 해소할 수 있도록
진입장벽을 낮출 수 있다.



프로젝트 소개

프로젝트명	Football Agora Project	
기획 목표	자신만의 풋살 스토리를 직접 기록하며 공유할 수 있는 장을 마련해주는 서비스 남녀노소 구분없이 누구든지 풋살 문화에 쉽게 접근할 수 있도록 풋살 특화 생성형 ai 서비스 제공	
투입 인원 및 소요 기간	총 8명 / 48일 (24.05.17~ 24.07.03)	
주요 기능	영상분석 팀 경기 유동 히트맵 선수별 볼 점유율 패스맵	풋살 가이드 챗봇 공식 한국 풋살 규정 안내 풋살 칼럼 (대회 안내, 부상 대처법 등) 국내 지역별 풋살장 위치 정보 제공 제작자(아디오스 팀) 정보 제공
Web 배포, 깃 허브	http://13.209.235.183/ , https://github.com/pladata-encore/DA34-final-Football_Agora.git	

Adios Team R&R

영상분석팀



김성일

영상 객체 인식 및 트래킹, 결과 가공,
모듈 통합, Git 관리



손명준

개인 단위 시각화, 코드 모듈화



임경란

기획, 팀 단위 시각화, 영상 전처리, 코드
모듈화

기획, DB모델링, AWS 배포



서한울

개발 팀



정현수

DB모델링, 웹서비스 개발, 모
델 서빙, Git 관리

LLM팀

기획, 데이터 전처리, Rag 파이프라인 구
축



강민지

LLM 모델 fine-tuning, 모델 서
빙, 튜닝 데이터 전처리, 크롤
링

모델 fine-tuning, 모델 서빙, 데이터 전처
리, 크롤링, 코드 모듈화, Git 관리



신대근



강동욱

A soccer ball is positioned on a green field, slightly to the right of the center. A white line runs diagonally across the field. In the background, a goalpost is visible under a clear sky. The overall scene is bright and sunny.

2

기획 및 추진 계획

프로젝트 진행 순서

기획

서비스 기획



요구사항 정리



분석 / 설계

데이터 정의 및 수집



WBS 작성



화면 설계



이행

핵심 기능 구현
(영상분석, 챗봇 모델)



Git 연동



개발환경 구축



ERD 설계 및 DB 구축



일반 기능 구현
(회원가입, 로그인 등)



AWS 서버 호스팅



서비스 배포

요구사항정의서

요구사항 정의서 (Requirment Definition)

*일반적으로는 메뉴그룹별, 화면 메뉴별로 요구사항 ID를 정의함

*요구사항의 수정, 추가, 삭제 사항을 상세히 기록하고, 요구사항 수정은 항상 프로젝트 변경관리 프로세스에 의하여 진행 / 수정 내용 알리는 날짜를 기록함

요구사항 ID	요구사항명	기능 ID	상세 요구사항명	상태	최초요청일자	최종수정일자	상세 설명	필수 데이터 요구사항	선택 데이터 요구사항	기타 요구사항
MEM01	로그인	MEM01_LOGIN01	자체 로그인 기능	최초	2024-05-22		FA 서비스 기능을 이용하기 위한 로그인	ID(또는 이메일), 비밀번호		
		MEM01_LOGIN02	카카오 연동 로그인 기능	최초	2024-05-22	2024-05-23	카카오 계정과 연동하여 아이디 및 비밀번호 입력 없이 쉽게 로그인 되도록 하는 기능			
		MEM01_LOGIN03	아이디 찾기	수정	2024-05-22	2024-05-28	로그인 시 아이디 찾는 기능	이름, 이메일		
		MEM01_LOGIN04	비밀번호 찾기	최초	2024-05-22	2024-05-28	로그인 시 비밀번호 재설정 기능	아이디, 이름, 이메일		
MEM02	로그아웃	MEM02_LOGOUT01	로그아웃	최초	2024-05-22		로그아웃 기능			
MEM03	회원가입	MEM03_SIGNUP01	자체 회원가입	최초	2024-05-22		사용자 입력 항목을 입력 받아 회원 가입하는 기능	아이디, 이메일, 비밀번호 (비밀번호 확인), 생년월일, 휴대폰 번호, 이름		
		MEM03_SIGNUP02	카카오 연동 회원가입	최초	2024-05-22	2024-05-23	카카오 계정과 연동하여 회원가입 하는 기능	카카오 아이디 (또는 이메일, 전화번호), 비밀번호		QR코드 로그인
		MEM03_SIGNUP03	회원가입 상세정보 입력	최초	2024-05-22		자체 회원가입 또는 카카오 연동 회원가입 기능 이후 사용자 정보 입력받는 기능	닉네임, 주활동지역, 플레이 포지션, 자신있는 능력, 플레이어 수준		
MEM04	회원 정보 수정	MEM04_MODIFY01	비밀번호 재설정	최초	2024-05-22			현재 비밀번호, 수정할 비밀번호		
		MEM04_MODIFY02	닉네임 재설정	최초	2024-05-22					
		MEM04_MODIFY03	프로필 사진	최초	2024-05-22				회원 프로필 사진	
		MEM04_MODIFY04	자기소개	최초	2024-05-22				자기소개 입력 사항	
		MEM04_MODIFY05	주활동지역 수정	최초	2024-05-22			주 활동지역 1, 2, 3		시, 구 형태로 입력 받음
		MEM04_MODIFY06	플레이 포지션 수정	최초	2024-05-22				공격, 수비, 미드, 골키퍼	
		MEM04_MODIFY07	자신있는 능력 수정	최초	2024-05-22				슛, 패스, 드리블, 체력, 스피드, 피지컬	최대 2개 중복선택 가능
		MEM04_MODIFY08	플레이어 수준 수정	최초	2024-05-22				루키, 스타터, 비기너, 아마추어, 세미프로, 프로	
		MEM04_MODIFY09	수정사항 저장	최초	2024-05-22					
		MEM04_MODIFY10	회원 탈퇴	최초	2024-05-22			비밀번호		

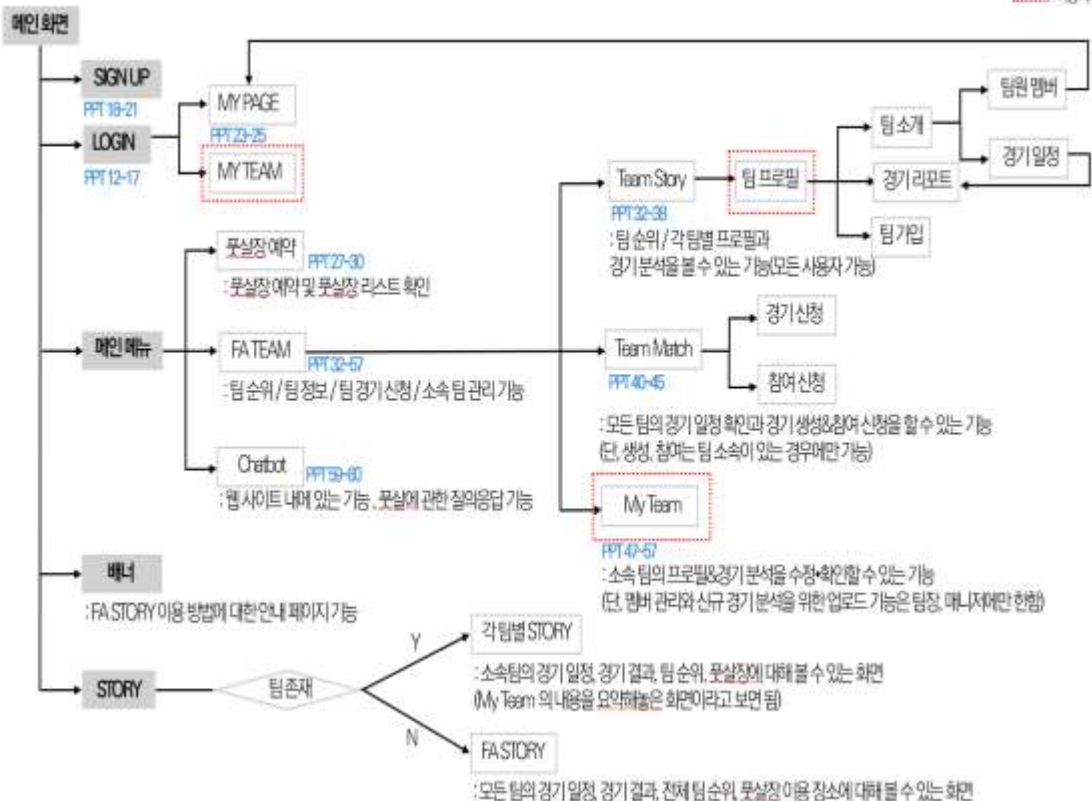
WBS

- 프로젝트 기간: 24.05.17~ 24.07.03

[illegible]

화면설계도

웹서비스구성화면요약본



화면명	대인용 화면_상단_로그인Ver
참고 요구사항 번호	
비고	작성자: 임경민

Football Agora

FATEAM

Chatbot

홍길동님

My Page

My Team

팀 생성은 어떻게 하나요?

배너

기능명세서

- 로그인시 개인이름으로 변경
- 축구 아이콘 활성화
- 축구 아이콘 클릭시 My Page, 팀관리 활성화
- 해당 클릭시 각 페이지로 이동
- 7초 & 3-1 클릭시 자동으로 배너 변경
- 각 배너 클릭시 해당 페이지로 이동
- FA TEAM의 STORY로 보이게
- Team Story) 팀 소개) 경기 일정에서 오늘 일자로 기준으로 경기 3시간 보아줌
- 초기 화면과 동일하게
- Team Story) 팀 소개로 페이지 이동
- 팀 가입하기 생성하기 모달창 띄우기
- Team Story) 팀 소개) 경기 일정에서 오늘 일자로 역순 기준으로 경기 3시간 보아줌
(경기 결과 업데이트 안 되었을 경우, vs 그대로 보아줌)
- 초기 화면과 동일
- Team Story) 팀 소개로 페이지 이동
- 팀 가입, 생성 모달창 띄우기

4 TEAM_A의 STORY

5 경기 일정

일시	팀 배정	장소
5월 22일 오후 8시	TEAM_A VS TEAM_B	서초아이경기장
5월 27일 오전 8시	TEAM_A VS TEAM_C	강남 경기장
5월 28일 오후 8시	TEAM_A VS 미정	서초아이경기장

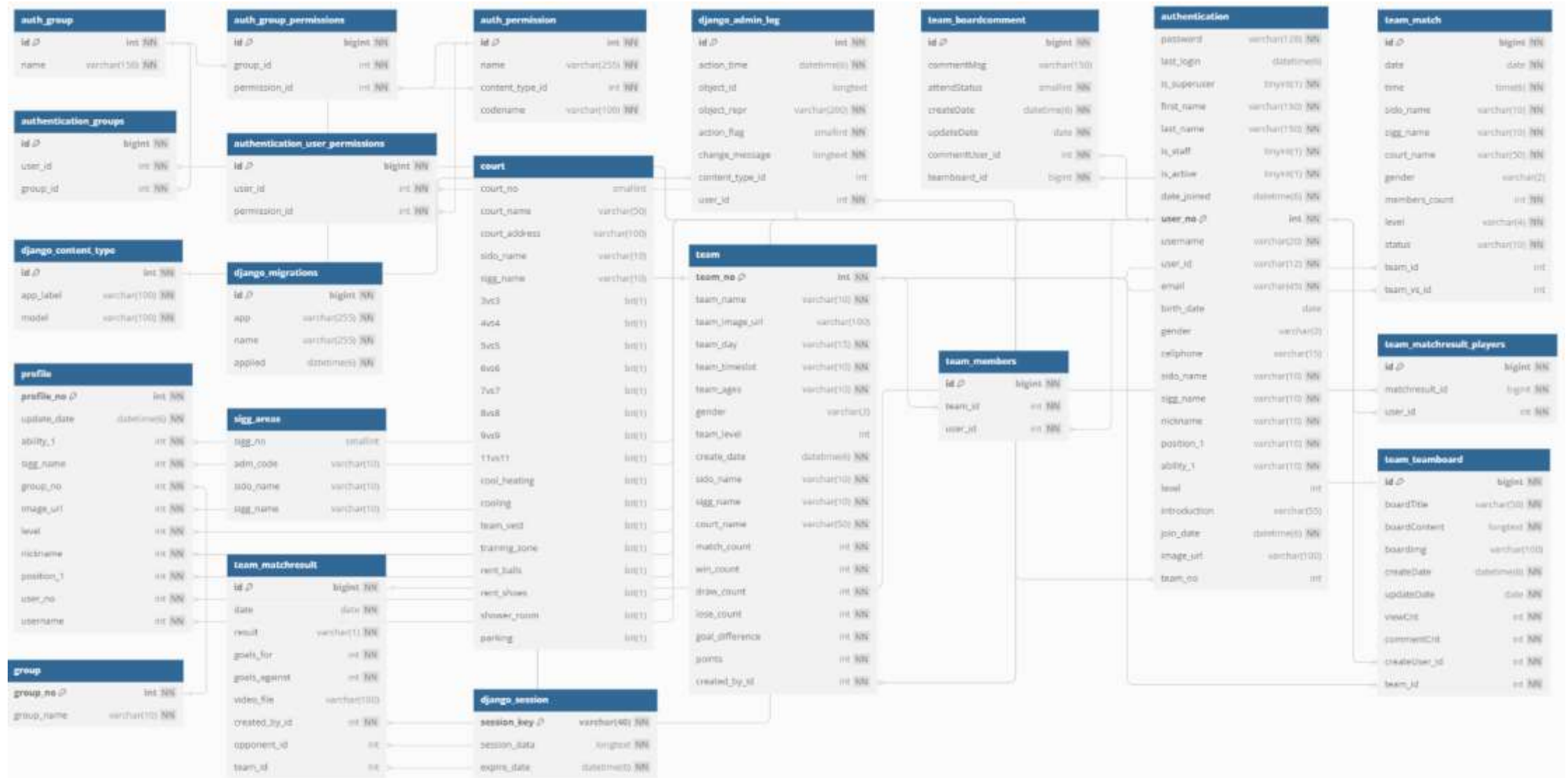
6+ 7 최근 경기 결과

일시	결과
5월 22일 오후 8시	TEAM_A 3-4 TEAM_B
5월 27일 오전 8시	TEAM_A 3-4 TEAM_B
5월 28일 오후 8시	TEAM_A 3-4 TEAM_B

8 팀 순위

순위	팀명	승	무	패	골득실	승점
1위	TEAM_A	1	1	1	1.6	1.6

ERD



개발 환경

#Front-end



#Back-end



#DB



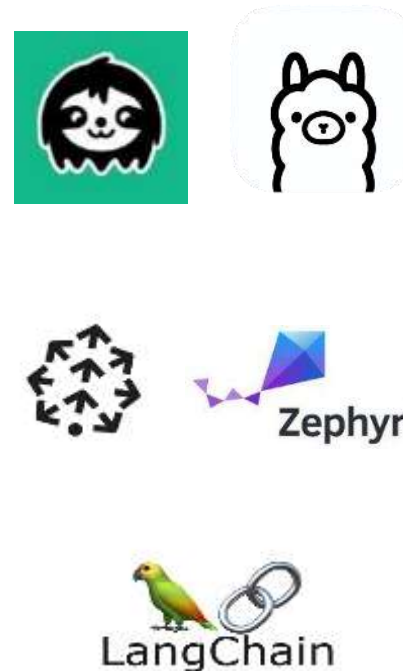
#Server



#image-DL



#NLP



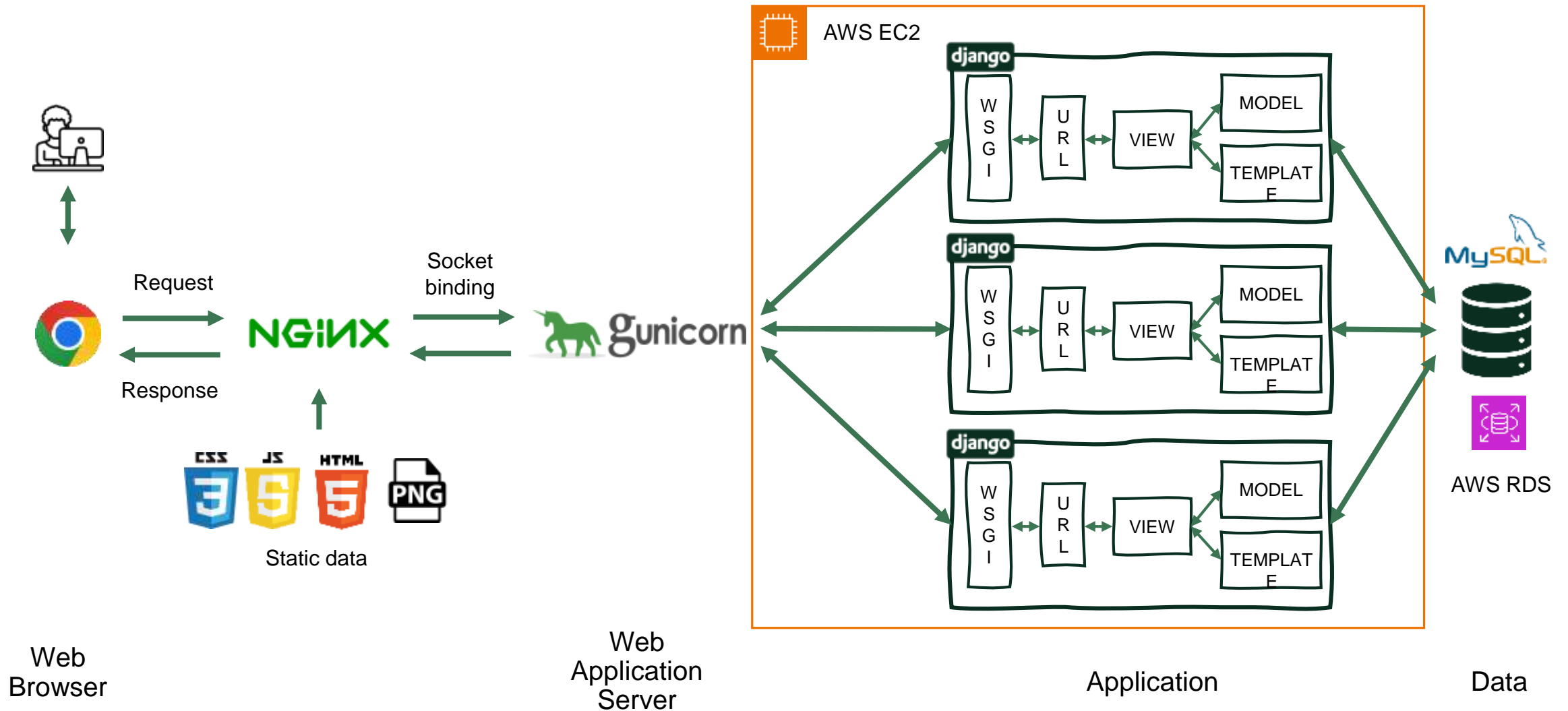
#Tool



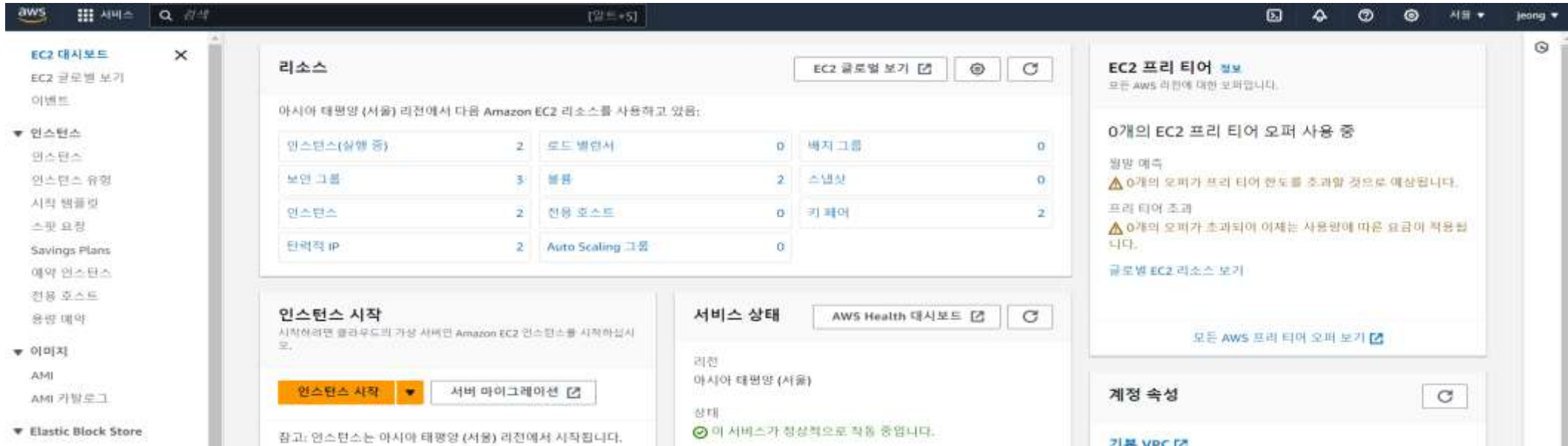
Django & Database 8

Client

Server



AWS instance



리소스

아시아 태평양 (서울) 리전에서 다음 Amazon EC2 리소스를 사용하고 있음:

리소스	수량	리소스	수량	리소스	수량
인스턴스(실행 중)	2	로드 밸런서	0	백지 그룹	0
보안 그룹	3	블록	2	스냅샷	0
인스턴스	2	전용 호스트	0	키 페어	2
탄력적 IP	2	Auto Scaling 그룹	0		

인스턴스 시작

시작하려면 클라우드의 가장 저렴한 Amazon EC2 인스턴스를 시작하십시오.

서비스 상태

리전: 아시아 태평양 (서울)

상태: ✔ 이 서비스가 정상적으로 작동 중입니다.

EC2 프리 티어

모든 AWS 리전에 대한 보편적입니다.

0개의 EC2 프리 티어 오퍼 사용 중

필만 예측

⚠ 0개의 오퍼가 프리 티어 한도를 초과할 것으로 예상됩니다.

프리 티어 초과

⚠ 0개의 오퍼가 초과되어 이제는 사용량에 따른 요금이 적용됩니다.

글로벌 EC2 리소스 보기

모든 AWS 프리 티어 오퍼 보기

계정 속성

기본 VPC

인스턴스 (2) 정보

인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

인스턴스 상태 = running X 필터 지우기

모든 상태 ▼

<input type="checkbox"/>	Name	인스턴스 ID	인스턴스 상태	인스턴스 유형	상태 검사	경보 상태	가용 영역	퍼블릭 IPv4 DNS	퍼블릭 IPv4 ...
<input type="checkbox"/>	FA_MainServer	i-0ca35c608bf009da6	실행 중	t2.micro	2/2개 검사 통과...	경보 보기 +	ap-northeast-2a		13.209.235.18:
<input type="checkbox"/>	FA_APIServer	i-0eeb2a7ede956f259	실행 중	g4dn.2xlarge	2/2개 검사 통과...	경보 보기 +	ap-northeast-2c		43.200.89.250

A soccer ball with blue and white patterns is positioned on a green grass field, resting on a white boundary line. In the background, a soccer goal is visible under a clear sky. A semi-transparent grey circle containing the number '3' is centered over the image.

3

개발 상세

개발 환경 구축

URL 정의

페이지	URL
메인화면	도메인/
로그인	도메인/login/
로그아웃	도메인/logout/
회원가입	도메인/signup/
회원탈퇴	도메인/delete_user/
My Page	도메인/mypage/
비밀번호 변경	도메인/password_change/
비밀번호 변경 완료	도메인/password_change/done/
아이디 찾기	도메인/find_id/
비밀번호 찾기(초기화)	도메인/password_reset/
비밀번호 찾기(확인)	도메인/reset/<uidb64>/<token>/
비밀번호 찾기(완료)	도메인/password_reset/done/

페이지	URL
챗봇	도메인/chatbot/

페이지	URL
My Team	도메인/myteam/
My Team(My Page에서 접속)	도메인/myteam/<int:team_id>/
팀 생성하기	도메인/create_team/
팀 가입하기	도메인/join_team/<int:team_id>/
팀 탈퇴(팀에 소속된 사용자)	도메인/leave_team/
팀 삭제(팀장)	도메인/delete_team/
팀 프로필 수정	도메인/update_team_profile/<int:pk>/
팀 리스트(팀이 없을때 보여지는 화면)	도메인/team_list/
Team Match 화면	도메인/matches/
매치 생성	도메인/matches/create/
경기 참여	도메인/matches/join/<int:match_id>/
경기 리포트 작성	도메인/match_report/<int:team_id>/
경기 리포트 수정	도메인/match_report_edit/<int:result_id>/
나의 팀 경기일정	도메인/match_schedule/
Team Story 화면	도메인/team_story/
영상분석	도메인/video_analysis/<int:team_id>/
팀 게시판 목록	도메인/board/<int:team_id>/
게시판 생성	도메인/board/<int:team_id>/create/
게시글 내용	도메인/board/<int:team_id>/<int:pk>/
게시글 수정	도메인/board/<int:team_id>/<int:pk>/update/
게시글 삭제	도메인/board/<int:team_id>/<int:pk>/delete/
댓글 삭제	도메인/board/<int:team_id>/<int:pk>/comment_delete/
댓글 수정	도메인/board/<int:team_id>/<int:pk>/comment_update/

개발 환경 구축

Django

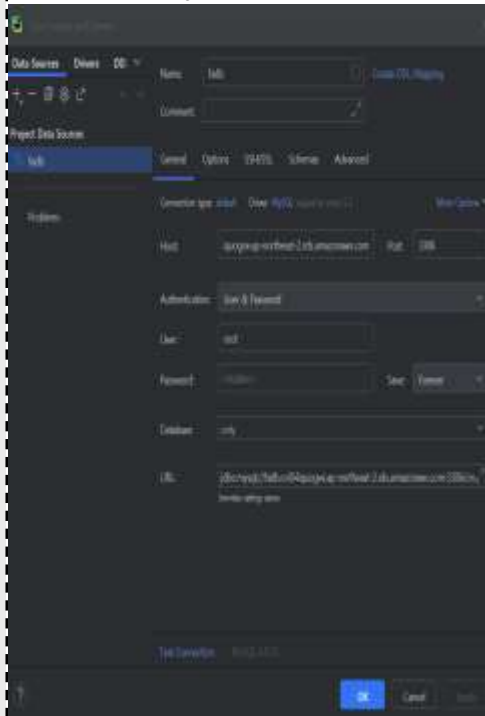
AWS
DB 구축



amazon
RDS



Pycharm



migrate

models.py로 변환

```
- python manage.py  
inspectdb >  
myapp/models.py
```

```
- python manage.py  
makemigrations
```

```
- python manage.py  
migrate
```

GET: 팀 순위, 경기 일정 및 결과

서비스 기능 구현

회원관리(인증,가입,탈퇴,개인정보업데이트)

활용 용도: 웹사이트의 기본적인 회원관리시스템

The image shows two side-by-side web forms. The left form is for password reset, titled 'Password Reset on Your Site Name'. It includes fields for '사용자 ID' (User ID) and '이메일주소' (Email address), and a '비밀번호 재설정' (Reset Password) button. The right form is for user registration, titled '회원가입' (Sign Up). It includes fields for '이름' (Name), '이메일주소' (Email address), '비밀번호' (Password), and '비밀번호 확인' (Confirm Password). Below these fields are several checkboxes for terms and conditions, and a '회원가입' (Sign Up) button.

The image shows a web form for user profile management. It includes a '회원가입' (Sign Up) button at the top. Below it are several sections: '회원가입' (Sign Up) with a '회원가입' button, '회원가입' (Sign Up) with a '회원가입' button, '회원가입' (Sign Up) with a '회원가입' button, and '회원가입' (Sign Up) with a '회원가입' button. The form also includes a '회원가입' (Sign Up) button at the bottom.

POST 요청: 인증, 가입, 탈퇴, 업데이트
DjangoPasswordResetView로 회원 이메일 인증

팀관리(팀생성및CRUD)

활용 용도: 사용자가 팀을 생성, 가입하고, 탈퇴, 수정, 등 팀내에서의 다양한 기능 제공

The image shows a web application for team management. It includes a '팀 관리' (Team Management) section with a '팀 생성' (Create Team) button. Below it is a '팀 목록' (Team List) table with columns for '순위' (Rank), '팀 이름' (Team Name), '경기' (Match), '승' (Win), '무' (Draw), '패' (Loss), '득점' (Goals), and '실점' (Goals Against). The table lists several teams, including 'FC 서울' (FC Seoul), 'FC 서울 U21', 'FC 서울 U18', 'FC 서울 U15', 'FC 서울 U12', 'FC 서울 U9', 'FC 서울 U7', 'FC 서울 U5', 'FC 서울 U3', 'FC 서울 U1', 'FC 서울 U0', 'FC 서울 U-1', 'FC 서울 U-2', 'FC 서울 U-3', 'FC 서울 U-4', 'FC 서울 U-5', 'FC 서울 U-6', 'FC 서울 U-7', 'FC 서울 U-8', 'FC 서울 U-9', 'FC 서울 U-10', 'FC 서울 U-11', 'FC 서울 U-12', 'FC 서울 U-13', 'FC 서울 U-14', 'FC 서울 U-15', 'FC 서울 U-16', 'FC 서울 U-17', 'FC 서울 U-18', 'FC 서울 U-19', 'FC 서울 U-20', 'FC 서울 U-21', 'FC 서울 U-22', 'FC 서울 U-23', 'FC 서울 U-24', 'FC 서울 U-25', 'FC 서울 U-26', 'FC 서울 U-27', 'FC 서울 U-28', 'FC 서울 U-29', 'FC 서울 U-30', 'FC 서울 U-31', 'FC 서울 U-32', 'FC 서울 U-33', 'FC 서울 U-34', 'FC 서울 U-35', 'FC 서울 U-36', 'FC 서울 U-37', 'FC 서울 U-38', 'FC 서울 U-39', 'FC 서울 U-40', 'FC 서울 U-41', 'FC 서울 U-42', 'FC 서울 U-43', 'FC 서울 U-44', 'FC 서울 U-45', 'FC 서울 U-46', 'FC 서울 U-47', 'FC 서울 U-48', 'FC 서울 U-49', 'FC 서울 U-50', 'FC 서울 U-51', 'FC 서울 U-52', 'FC 서울 U-53', 'FC 서울 U-54', 'FC 서울 U-55', 'FC 서울 U-56', 'FC 서울 U-57', 'FC 서울 U-58', 'FC 서울 U-59', 'FC 서울 U-60', 'FC 서울 U-61', 'FC 서울 U-62', 'FC 서울 U-63', 'FC 서울 U-64', 'FC 서울 U-65', 'FC 서울 U-66', 'FC 서울 U-67', 'FC 서울 U-68', 'FC 서울 U-69', 'FC 서울 U-70', 'FC 서울 U-71', 'FC 서울 U-72', 'FC 서울 U-73', 'FC 서울 U-74', 'FC 서울 U-75', 'FC 서울 U-76', 'FC 서울 U-77', 'FC 서울 U-78', 'FC 서울 U-79', 'FC 서울 U-80', 'FC 서울 U-81', 'FC 서울 U-82', 'FC 서울 U-83', 'FC 서울 U-84', 'FC 서울 U-85', 'FC 서울 U-86', 'FC 서울 U-87', 'FC 서울 U-88', 'FC 서울 U-89', 'FC 서울 U-90', 'FC 서울 U-91', 'FC 서울 U-92', 'FC 서울 U-93', 'FC 서울 U-94', 'FC 서울 U-95', 'FC 서울 U-96', 'FC 서울 U-97', 'FC 서울 U-98', 'FC 서울 U-99', 'FC 서울 U-100'. Below the table is a '팀 생성' (Create Team) button.

POST: 팀 가입, 탈퇴, 삭제, 팀 프로파일 수정
GET: 팀 목록 조회

서비스 기능 구현

경기매칭생성및결과업데이트

활용 용도: 경기 매칭을 성사하고, 경기 결과를 바탕으로 FA팀들 순위 업데이트

경기 목록		경기 생성됨 이름: 바르셀로나		경기 생성됨 이름: 바르셀로나		경기 생성됨 이름: 바르셀로나	
일자: 2024년 6월 22일 오후 8:00		일자: 2024년 6월 22일 오후 8:00		일자: 2024년 6월 22일 오후 8:00		일자: 2024년 6월 22일 오후 8:00	
장소: 세르도르스디아로 스타디움		장소: 세르도르스디아로 스타디움		장소: 세르도르스디아로 스타디움		장소: 세르도르스디아로 스타디움	
상대: 바르셀로나		상대: 바르셀로나		상대: 바르셀로나		상대: 바르셀로나	
인원: 99,000		인원: 99,000		인원: 99,000		인원: 99,000	
수중: 30%		수중: 30%		수중: 30%		수중: 30%	
부상: 1명		부상: 1명		부상: 1명		부상: 1명	
경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1	
경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드	
일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00	
장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움	
상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드	
인원: 99,000		인원: 99,000		인원: 99,000		인원: 99,000	
수중: 30%		수중: 30%		수중: 30%		수중: 30%	
부상: 1명		부상: 1명		부상: 1명		부상: 1명	
경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1	
경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드	
일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00	
장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움	
상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드	
인원: 99,000		인원: 99,000		인원: 99,000		인원: 99,000	
수중: 30%		수중: 30%		수중: 30%		수중: 30%	
부상: 1명		부상: 1명		부상: 1명		부상: 1명	
경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1		경기 결과: 2-1	
경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드		경기 생성됨 이름: 레알 마드리드	
일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00		일자: 2024년 6월 20일 오후 8:00	
장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움		장소: 구르니아르FC 스타디움	
상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드		상대: 레알 마드리드	
인원: 99,000		인원: 99,000		인원: 99,000			

POST: 경기참여, 경기생성, 경기 리포트 작성
GET: 팀 순위, 경기 일정 및 결과

팀게시판

활용 용도: 경기에 참여할 팀멤버를 구하고, 같은 팀원끼리 소통의 장 마련

6/25 경기 구합니다

6/25 도곡 회고 풋살장 19:00 5대5 혼성매칭 하실 분 구합니다!!! 매치 수준: 풋야마

작성자: messi | 작성일: 2024년 7월 1일 10:06 오후 | 조회수: 0

수정 삭제

댓글

messi: 참석이요!

참석 여부: 참석

수정 삭제

댓글 작성

CommentMsg:

참석 여부:

☐ 참석
 ☐ 불참석
 ☐ 보류

댓글 작성

게시판 목록

게시글 작성

7/2 안양대학교 20:00 남자 5대5 매치 하실 분 구합니다! 2024년 7월 1일 10:08 오후

6/25 경기 구합니다 2024년 7월 1일 10:06 오후

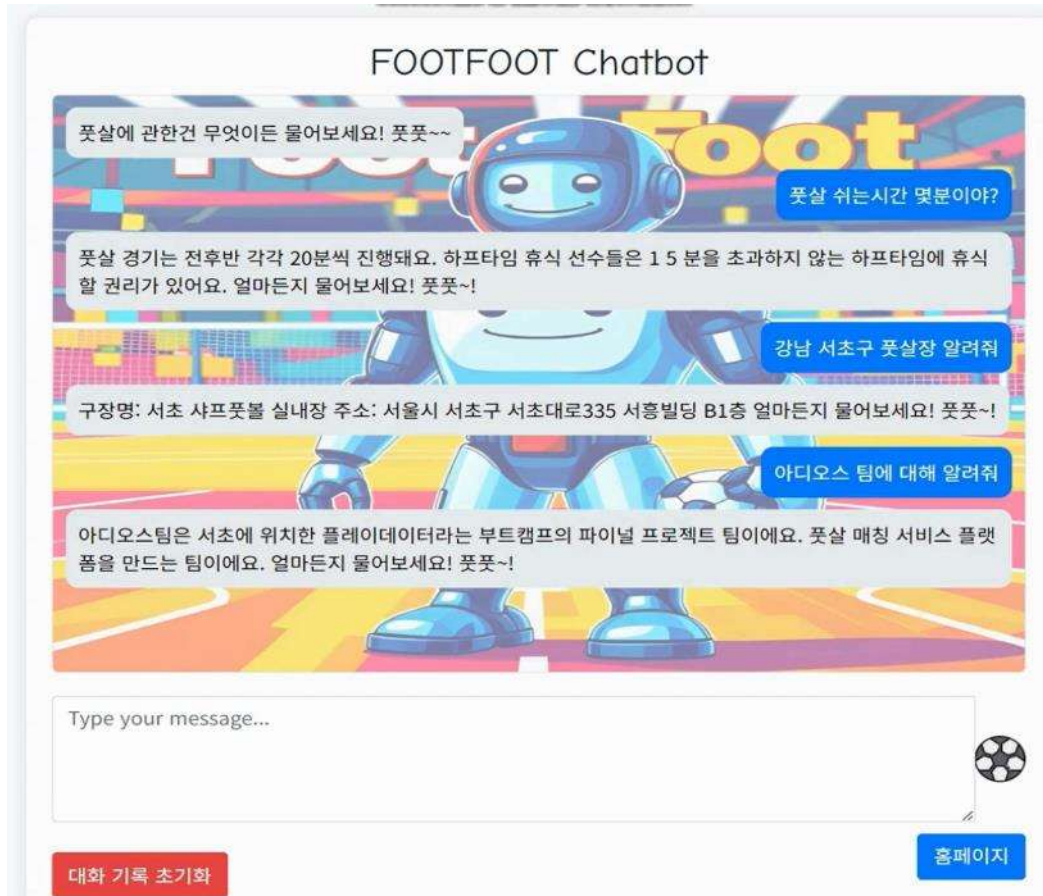
노노노 2024년 6월 25일 6:21 오후

POST: 게시물 작성, 댓글 작성, 참석여부, 수정, 삭제
GET: 게시판 목록

서비스 기능 구현

FA서비스이용관련챗봇

활용 용도: 풋살에 접근성을 높이기 위한 챗봇



FastApi를 이용한 챗봇 모델 서빙, 화면 구축

RSS:최신해외축구정보제공

활용 용도: 사용자의 흥미를 돋을 해외축구기사 제공

해외축구 기사

'Who else? Iconic Bellingham goal will go down in England history'

Sun, 30 Jun 2024 21:33:32 GMT

From 'Iceland vibes' to 'get-out-of-jail-free card'

Sun, 30 Jun 2024 20:52:16 GMT

Bellingham's last-gasp equaliser - All the angles

Sun, 30 Jun 2024 21:27:18 GMT

'A little bit desperate' - Norris fumes at Verstappen move

Sun, 30 Jun 2024 18:29:49 GMT

Highlights: England edge past Slovakia in extra time

Sun, 30 Jun 2024 18:38:27 GMT

Feedparser 모듈을 사용해 BBC 해외축구 기사 링크 연결

홈페이지 영상

경기결과분석리포트(경기녹화영상기반)

팀 경기 영상 분석 - 정현수의 팀

날짜 선택: 2024년 6월 15일 선택

2024년 6월 15일: 정현수의 팀 vs 신대근의 팀

결과: Lose

득점: 1

실점: 2

비디오 파일:

왼쪽 비디오:

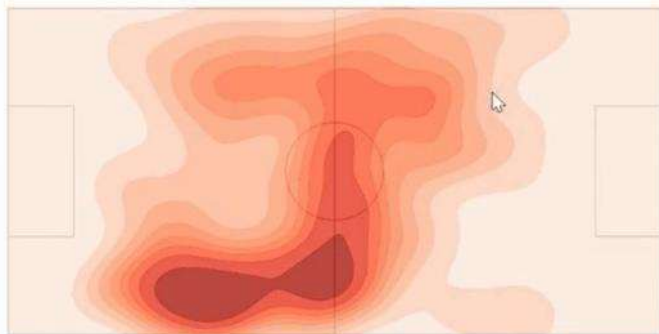


오른쪽 비디오:

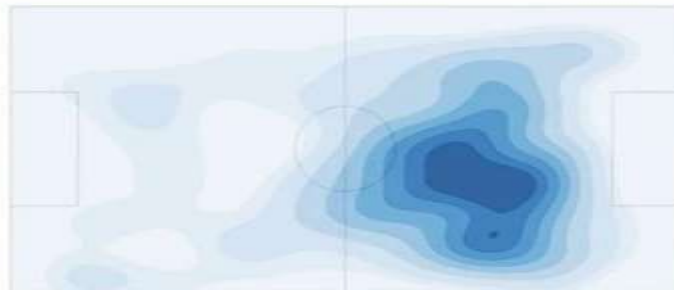


분석 결과

Heatmap Home:



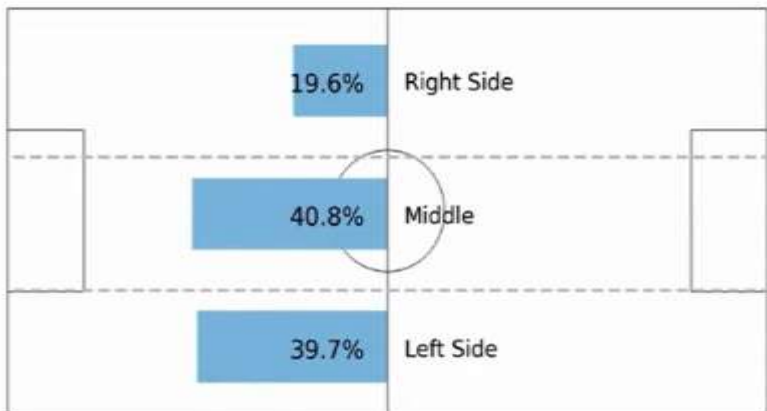
Heatmap Away:



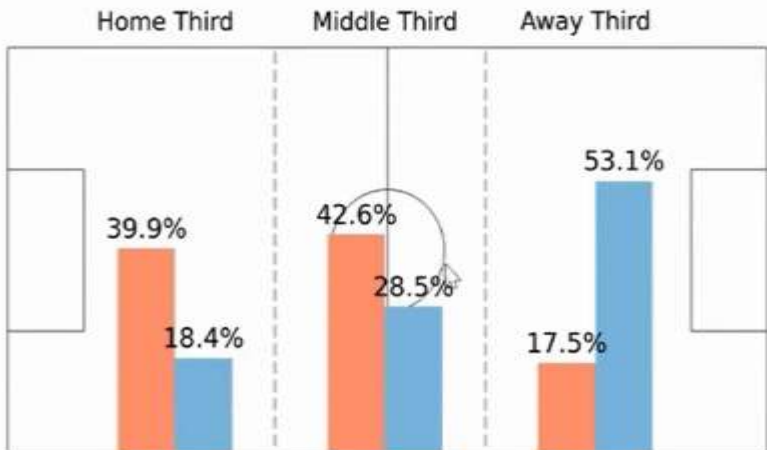
Hasball LMR Home:



Hasball LMR Away:



Hasball DMR:



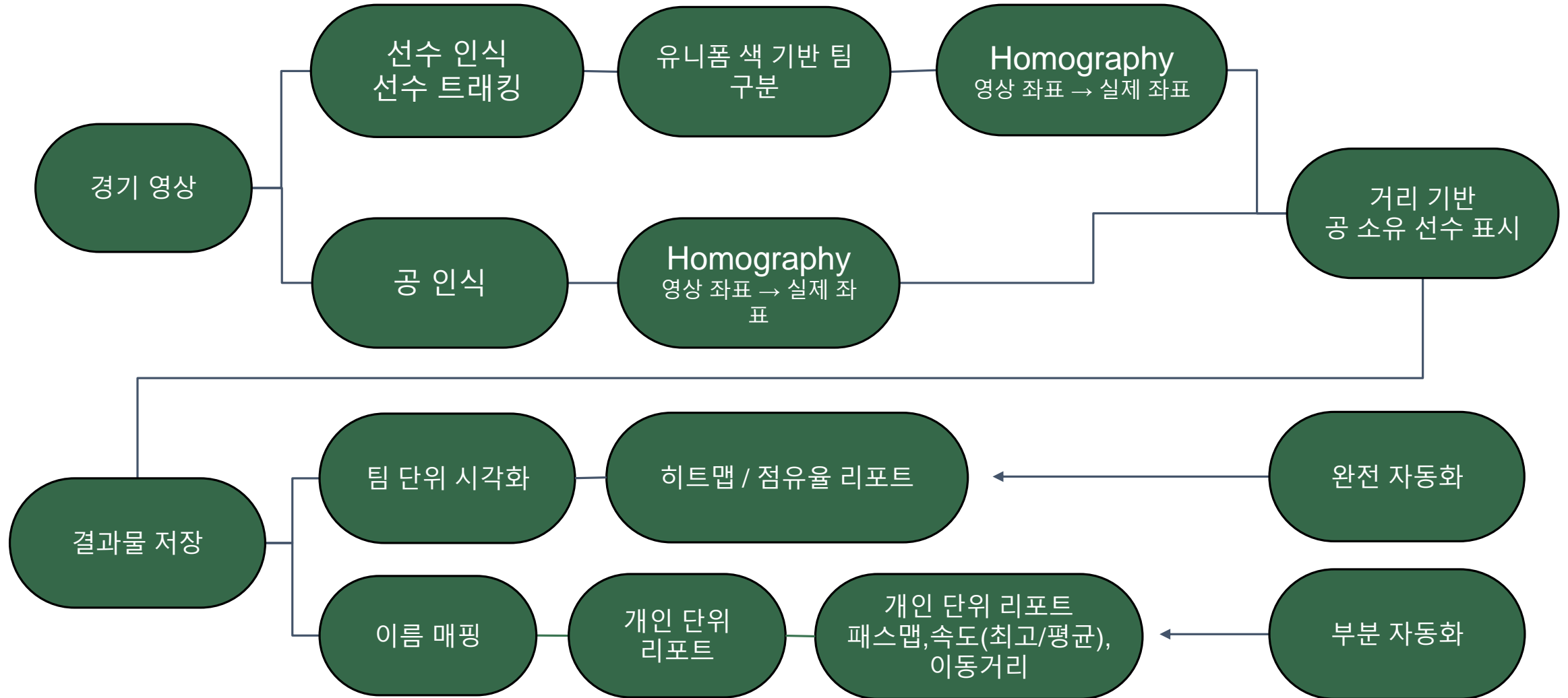
영상 분석 시작

A soccer ball with blue and white patterns is positioned on a green grass field, resting on a white boundary line. In the background, a soccer goal is visible under a clear sky. The scene is bathed in soft, warm light, suggesting late afternoon or early morning.

4

주요 기능

영상분석 FLOW



객체 인식 및 트래킹

1 객체 인식

선수 인식



→ 모델이 가볍고 성능이 준수함

공 인식

YOLOv5

→ 작은 객체에 대한 검출력이
 좋음

2 트래킹

선수 트래킹 : ByteTracker

[객체인식 + 트래킹 테스트 결과 영상 링크](#)

유니폼 색 기반 팀 구분

1. Input 영상의 첫 번째 frame의 bounding-box 기준으로 track-id의 이미지 추출
2. 선수 이미지의 중앙부분만 crop
3. crop한 이미지의 색상 코드를 RGB → Lab으로 변경 (배경과 유니폼 색의 명확한 구분을 위함)

4. **KMeans Clustering** 모델을 활용해 두 개의 클러스터로 전환

5. 많은 수의 클러스터를 유니폼 색상으로 부여

6. 유니폼 색깔들을 하나의 array에 삽입

7. **KMeans Clustering** 모델을 활용해 유니폼 색들을 두 개의 클러스터로 분류

8. 각 프레임 별로 유니폼 색 추출

9. 7의 클러스터링 모델에 8의 유니폼 색 predict → 선수의 소속 팀

Original Image



Cropped Image



LAB Image



Clustered Image



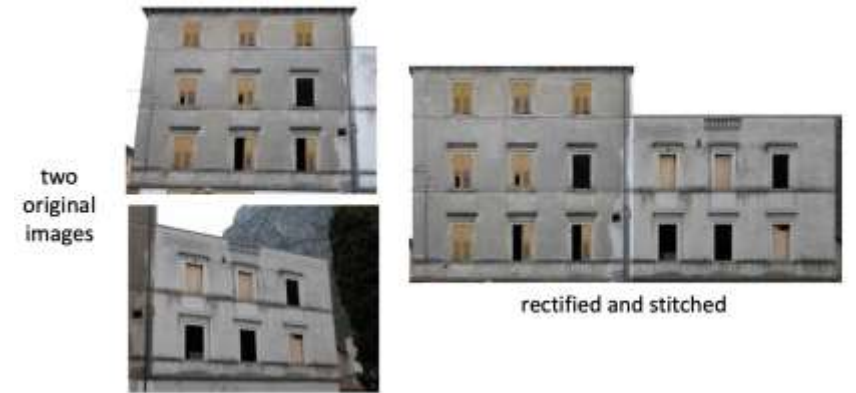
Image Homography

[Image Homography]

: 3D 공간에서 서로 다른 두 시점에서 바라본 이미지를 다른 각도로 변환하는 기술

[실제 영상 적용 방법]

1. 왼쪽, 오른쪽 경기장 촬영 영상에서 경기장 선의 교차점, 코너 부분을 keypoint로 설정
2. 실제 크기 비율의 경기장 라인 사진의 keypoint와 영상의 keypoint를 매치
→ 각각의 영상에 대한 Homography Matrix 생성
3. 선수와 공의 bounding-box에서 밑변의 중점을 frame 상의 좌표로 고정
4. frame 상의 좌표에 Homography Matrix 연산 → `cv2.perspectiveTransform()`
5. 왼쪽 경기장, 오른쪽 경기장 영상에서 변환된 좌표가 실제 중앙 선을 넘는다면 삭제



Issue - Tracking

1. track-id switching

한 화면 내에서 두 선수가 겹치게 될 때, 두 선수의 track-id가 바뀜



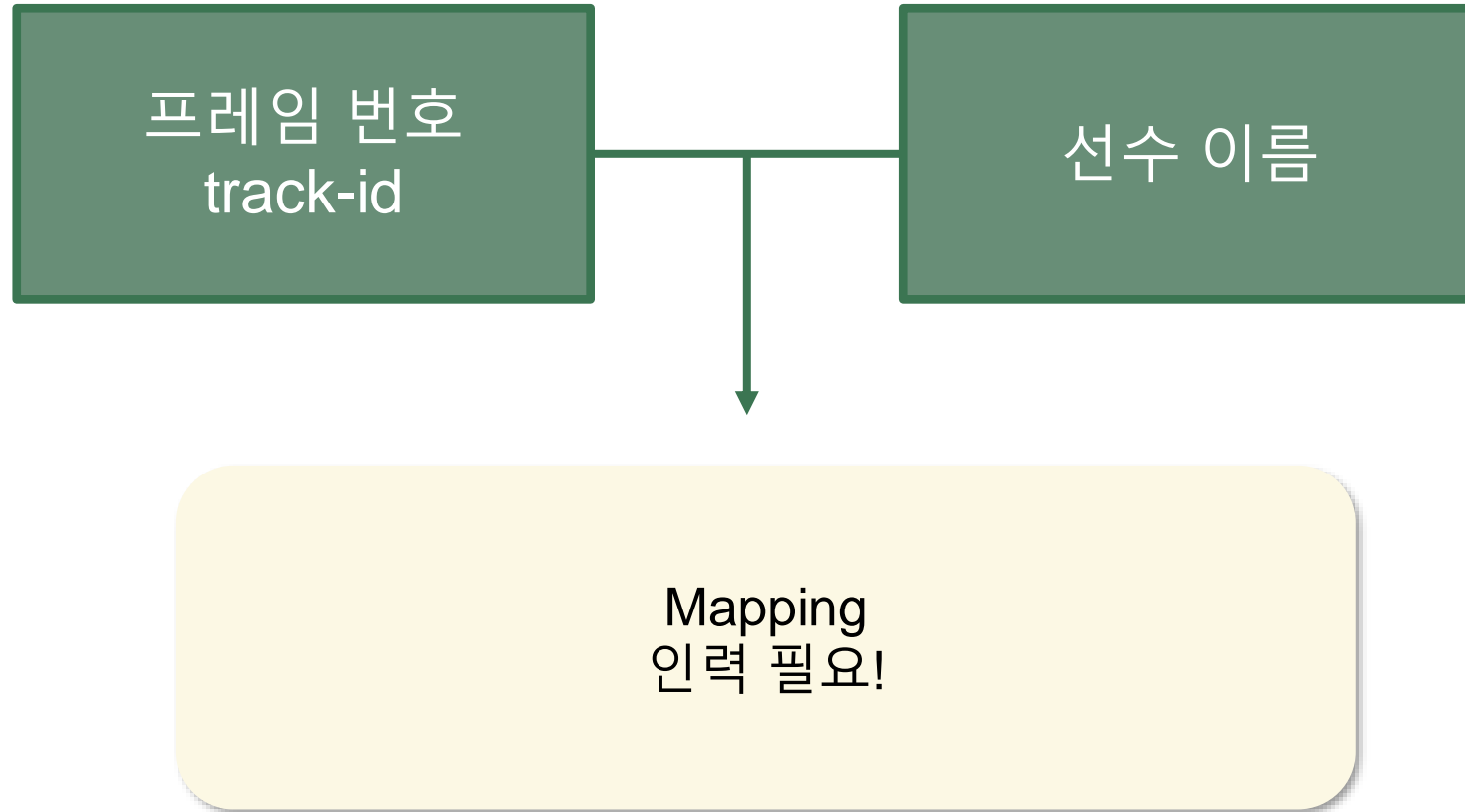
2. track-id lost

tracking을 할 때, 추적한 객체가 화면 밖으로 나가면 새로운 track-id를 부여함

한 명의 선수에 대해 여러 개의 track-id가 생성됨

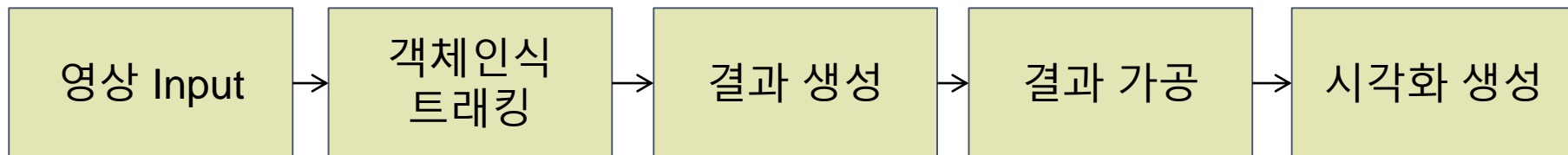


Solution - Tracking



Solution - Tracking

1 초기 진행 방향



2 변경사항



팀단위 리포트 - 유동 히트맵

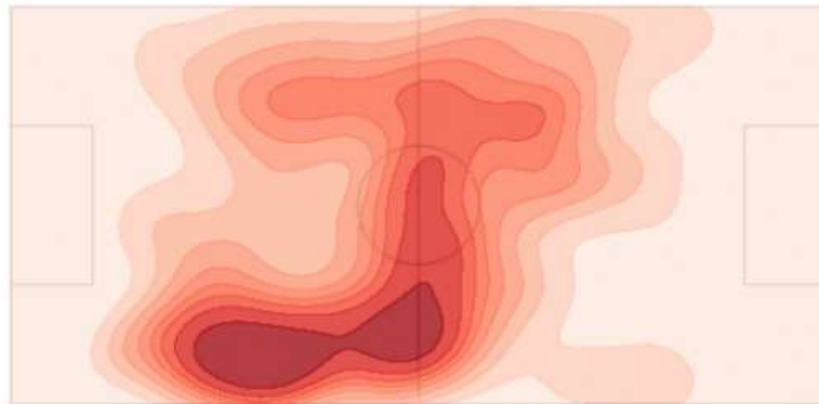
frame_num	player_id	team	team_color	has_ball	coord_x	coord_y	start_pitch_side	in_pa
0	0	OJY	1	[230, 61, 37]	True	368	212	left non_pa
8	1	OJY	1	[230, 61, 37]	True	368	211	left non_pa
16	2	OJY	1	[230, 61, 37]	True	371	213	left non_pa
24	3	OJY	1	[230, 61, 37]	True	376	216	left non_pa
32	4	OJY	1	[230, 61, 37]	True	378	216	left non_pa
...
9117	1139	OJY	1	[230, 61, 37]	True	575	28	left non_pa
9125	1140	OJY	1	[230, 61, 37]	True	575	27	left non_pa
9289	1161	SDG	1	[230, 61, 37]	True	654	64	left non_pa
9313	1164	SDG	1	[230, 61, 37]	True	662	66	left non_pa
9337	1167	SDG	1	[230, 61, 37]	True	631	90	left non_pa

- 영상 객체 인식을 통한 stub.pkl 의 데이터 프레임

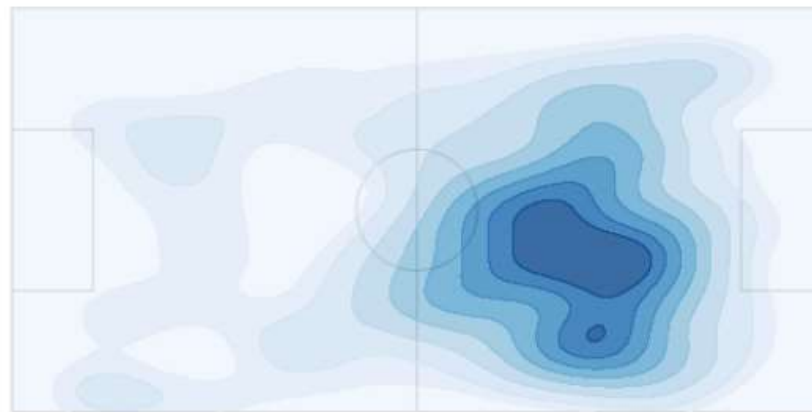
```
if team=='Team-A':  
    # KDE 히트맵 생성  
    sns.kdeplot(  
        x=df.query("in_pa!='left_pa')["coord_x"],  
        y=df.query("in_pa!='left_pa')["coord_y"],  
        fill=True,  
        thresh=0,  
        levels=10,  
        cmap="Reds",  
        alpha=0.8,  
        ax=ax)
```

- 유동 히트맵 시각화 방법

- 1) 골키퍼를 제외한 선수들에 대한 유동 히트맵 시각화
- 2) kde 히트맵 사용
 - 확률 밀도추정분포를 사용함으로써, 밀집 지역을 쉽게 시각화 가능



TEAM A 유동 히트맵



TEAM B 유동 히트맵

팀단위 리포트 - 점유율

```
team_a = pd.DataFrame({
    'Y_Proportion': team_a_y.values,
    'X_Proportion': team_a_x.values
})

team_b = pd.DataFrame({
    'Y_Proportion': team_b_y.values,
    'X_Proportion': team_b_x.values
})

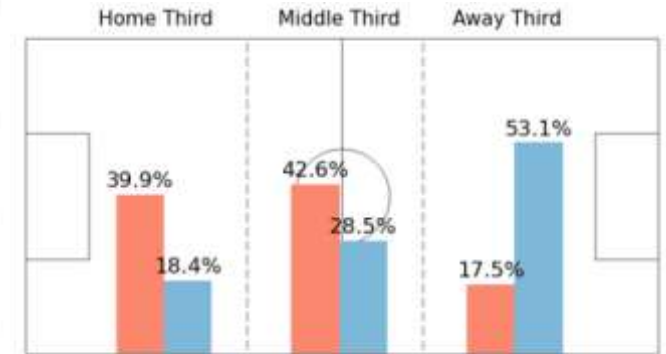
team_a
```

	Y_Proportion	X_Proportion
0	35.426009	39.910314
1	42.376682	42.600897
2	22.197309	17.488789

```
team_b
```

	Y_Proportion	X_Proportion
0	19.553073	18.435754
1	40.782123	28.491620
2	39.664804	53.072626

구역별 점유율



- 점유율 시각화 방법

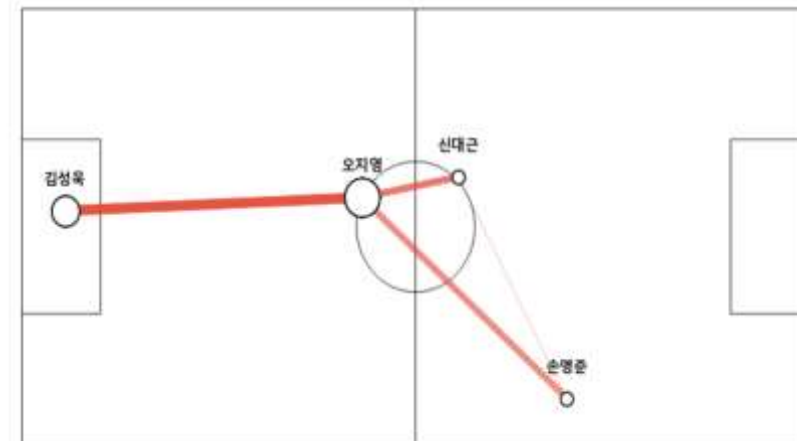
- 1) has_ball(공 소유)에 해당하는 값만 기준으로 삼음
- 2) 각 팀별 coord_x, y 좌표 값의 max값을 기준으로 3등분으로 나눠 구역별 점유율 시각화

개인단위 리포트 - 패스맵

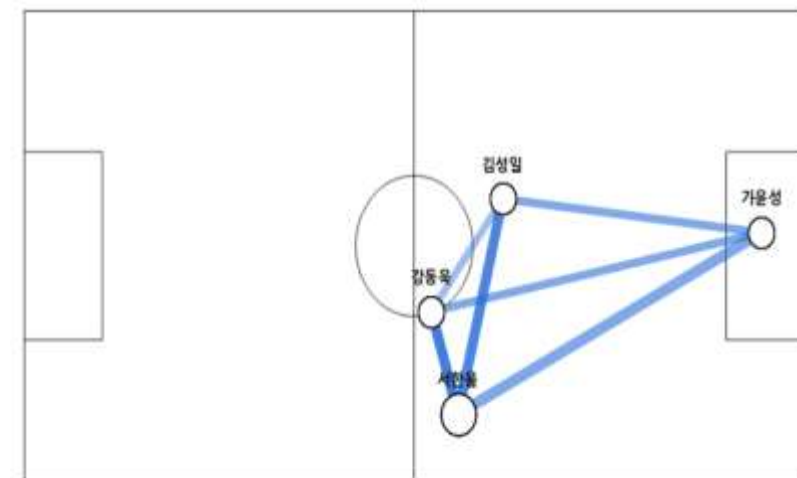
패스맵 시각화 방법

- 1) has_ball = True에 해당하는 값만 기준으로 삼음
- 2) passer와 receiver가 같은 팀이면 pass_success로 count
- 3) pass의 성공 횟수에 따라 원 크기와 선 색깔 구별지음

frame	passer_name	passer_initial	passer_coord	receiver_name	receiver_initial	receiver_coord	team
9	오지영	OJY	[383, 201]	신대근	SDG	[431, 7]	1
48	김성일	KSI	[610, 81]	가윤성	KYS	[732, 130]	0
101	가윤성	KYS	[751, 216]	서한울	SHW	[633, 233]	0
111	서한울	SHW	[612, 243]	김성일	KSI	[419, 183]	0
248	서한울	SHW	[591, 342]	강동욱	KDW	[602, 313]	0
257	강동욱	KDW	[602, 301]	김성일	KSI	[611, 163]	0



TEAM A 패스맵



TEAM B 패스맵

개인단위 리포트 - Stats

```
for object, object_tracks in tracks.items():
    if object == "ball":
        continue
    number_of_frames = len(object_tracks)
    for frame_num in range(0, number_of_frames, self.frame_window):
        last_frame = min(frame_num + self.frame_window, number_of_frames - 1)

        for track_id, _ in object_tracks[frame_num].items():
            if track_id not in object_tracks[last_frame]:
                continue

            start_position = object_tracks[frame_num][track_id]['coord_tr']
            end_position = object_tracks[last_frame][track_id]['coord_tr']

            if start_position is None or end_position is None:
                continue

            # 픽셀 단위를 실제 거리 단위로 변환
            distance_covered_x = (end_position[0] - start_position[0]) / 40
            distance_covered_y = (end_position[1] - start_position[1]) / 40
            distance_covered = (distance_covered_x**2 + distance_covered_y**2)**0.5

            time_elapsed = (last_frame - frame_num) / self.frame_rate
            speed_meters_per_second = distance_covered / time_elapsed
            speed_km_per_hour = (speed_meters_per_second) * 3.6
```

	player_name	team	average_speed(km/h)	max_speed(km/h)	total_distance(m)	pass_success_count	pass_fail_count	pass_intercept_count	pass_success_rate(%)
0	오지영	1	5.973237168375033	21.148957421111806	199.61	8	7	8	53.33
1	손명준	1	5.42673192727364	11.476166607365021	181.34	1	5	3	16.67
2	김성욱	1	1.1230800978677355	7.903619424036054	37.53	5	1	4	83.33
3	신대근	1	7.367830798027577	26.98693016999155	246.21	1	10	7	9.09
4	서한울	0	4.976710048821218	25.615151766093437	166.31	9	4	6	69.23
5	김성일	0	6.1920914511503	35.53722555293252	206.92	5	10	9	33.33
6	강동욱	0	4.936109930538851	12.310970717209914	164.95	5	7	6	41.67
7	가윤성	0	2.6114784291256665	34.46152637362425	87.27	5	1	2	83.33

개인 스탯 리포트

개인스탯리포트 시각화 방법

- 1) 객체와 트랙 정보를 이용하여 주어진 프레임 윈도우 내에서 속도와 이동거리를 계산
- 2) 바운딩 박스를 이용해 객체의 시작과 끝 위치를 구하고, 이를 실제거리단위로 변환한 후 속도와 거리를 계산하여 트랙에 추가

LLM Fine-tuning & RAG

이름 : 풋풋이

풋살 초보자를 위한 친절한 챗봇

말투 : '해요'체를 쓰며 말끝에 '풋풋~!'을 사용



<예시 화면>

Fine-Tuning vs RAG

방식 비교

Fine-tuning

- 특정 도메인에 대해 깊이 있고 일관된 응답이 필요한 경우.
- 충분한 양의 도메인 특화 데이터가 확보된 경우.
- 정보의 일관성과 응답 속도가 중요한 경우.
- 모델이 자주 업데이트될 필요가 없는 경우.

VS

RAG

- 최신 정보가 중요하고, 자주 업데이트가 필요한 경우
- 다양한 출처에서 정보를 가져와야 하는 경우
- 모델 파라미터 튜닝 없이 성능 향상을 원하는 경우
- 빠르게 새로운 정보나 데이터를 통합해야 하는 경우.



한국어 '생성 능력' 향상. 곳곳이 컨셉 말투 학습을 위해 Fine-Tuning 우선 진행

LLM Fine-tuning

Fine-tuning

Preparation

Pre-trained



Dataset



Fine-tuning

LoRA(Low Rank Adaptation)

```
from trl import SFTTrainer
from transformers import TrainingArguments

model = FastLanguageModel.get_peft_model(
    model,
    r=32,
    lora_alpha=64,
    lora_dropout=0.05,
    target_modules=[
        "q_proj", # Query projection
        "k_proj", # Key projection
        "v_proj", # Value projection
        "o_proj", # Output projection
        "gate_proj", # Gate projection
        "up_proj", # Upstream projection
        "down_proj", # Downstream projection
    ],
    bias="none",
    use_gradient_checkpointing="unsloth",
    random_state=123,
    use_rslora=False,
    loftq_config=None,
)

tokenizer.padding_side = "right"

trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    train_dataset=dataset,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    dataset_num_proc=2, # 데이터 처리에 사용할 프로세스 수
    packing=False, # 짧은 시퀀스에 대한 학습 속도를 가속시킬 수 있음
    args=TrainingArguments(
        per_device_train_batch_size=a,
```

Model

.gguf / .ggml



Safetensors



LLM Fine-tuning data

Dongwookss/
q_a_korean_
futsal

nlpai-lab/databricks-dolly-15k-ko
데이터셋 15000개를 베이스로 구축
자체 제작 데이터추가 후 '해요체' 변경
뜻뜻이 컨셉 말투 붙여 완성

mintaeng/
llm_futsaldata_yo

beomi/KoAlpaca-v1.1a
데이터셋 21200개 베이스로 구축
q_a korean futsal 데이터 추가
'해요체' 변경

instruction string - lengths	input float64	output string - lengths
5~1.05k 100%		15~466 88.4%
토성의 고리가 빛의 띠로 보이는 이유는 무엇인가요? 토성의 고리는 얼음과 같은 여러 물질로 이루어져 있다고 알고 있는데, 카시니가 찍은 사진에서 마치 빛의 띠처럼 보이는 이유가 무엇인가요? 물질의 공전 속도가 빠르...	null	토성의 고리가 미세한 입자들로 이루어져 있기 때문에, 입자들의 밀도 차이 때문에 카시니 탐사선에서 찍은 고 해상도 사진에서 빛의 띠가 보이는 것인데요. 실제로는 토성의 고리 입자들의 운동이 장노출 사진에서 잔상이...
화장품 OEM과 화장품 ODM의 차이점은 무엇인가요? 화장품 자체 제조 브랜드 런칭을 위해 OEM과 ODM용어에 대해 혼란스러움을 느끼고 있습니다. 두 용어의 차이점이 무엇인지 알고 싶습니다.	null	화장품 제조업체는 대체로 OEM과 ODM을 통해 제품을 만들어요. OEM은 브랜드에서 제품을 주문하였을 때, 반제품이나 완제품으로 납품받는 방식이에요. 반면 ODM은 자체 개발 능력을 갖춘 제조원이 유종까지 담당하여 상품...
'사이보그'는 언제 처음 등장한 말이며, 그 의미와 종류에는 어떤 것이 있는지 알고 싶습니다.	null	'사이보그'는 1960년에 처음 등장한 말로, 기계와 유기체가 합성되어 생겨난 새로운 개념을 나타내요. 주로 공상과학영화에서 등장하며, 1985년 페미니스트인 도나 해러웨이가 발표한 '사이보그를 위한 선언문'으로 인해...
씨앗을 냉장 처리후 심는 이유와 장기보존하려면 어떻게 해야 할까요?	null	1. 씨앗을 발아과정에서 냉장실에 보관후 심는 이유는 씨앗에는 휴면기간이 있기 때문이에요. 몇 년 동안도 휴면기간이 있는 경우가 있으며, 이러한 경우 계절의 변화 등을 거치며 발아할 준비가 돼요. 이러한 식물 중 대표적으로 잣나무가 있으며, 노천매장을 할 경우 1~2년 후에 발아하기 때문에 썩지 않도록 이를 방지하기 위해 냉장보관 후 심을 수 있어요. 2. 지금 당장 심지 않을 씨앗을 장기보존하려면, 식물의 종류에 따라 냉장보관 또는 냉동보관이 가능해요. 대부분의 경우 냉장보관이 가능하지만, 일부의 식물은 냉동보관이 필요해요. 보관하기 전에 씨앗을 완전히 건조시켜야 하며, 습습지 장소에 보관하거나 실온보다는 낮은 온도(2~10도)에서 보관하는 것이 좋아요. 이렇게 보관했을 경우 씨앗의 보존기간은 상대적으로 길어지게 되며, 필요할 때마다 사용할 수 있어요. 얼마든지 물어보세요! 꾀꾀~!
야구 글러브의 종류와 포지션별 특징은 무엇인가요?	null	보통 4칸으로 그물처럼 되어있는 것이 내야글러브이며, 1루를 제외한 내야 전 포지션에서 사용 가능해요. 그물처럼 촘촘히 되어있는 것이 외야글러브이며, 투수수, 중견수, 좌익수에서 사용 가능해요. 투수 글러브는 그림...
OPEC+와 OPEC의 차이점은 무엇인가요?	null	석유 수출 기구(OPEC)와 러시아를 10개 주요 산유국의 연합체(OPEC+)가 있어요. OPEC은 1960년 9월 이라크, 이란, 쿠웨이트, 사우디아라비아, 베네수엘라가 바그다드에서 창설한 국제기구이며, 현재 이라크, 이란, 쿠웨...
광해군은 폭군이었나요? 그 업적은 어떤가요?	null	광해군이 폭군인가에 대해서는 서인, 인조, 인목대비 등의 시각에서는 폭군이라고 말하지만, 역사적으로는 패자가이기 때문에 그들의 시각만 볼 수는 없어요. 광해군의 업적으로는 산림 숭흥, 실리 외교, 경기도 대풍법 시행...

LLM Fine-tuning

```
from trl import SFTTrainer
from transformers import TrainingArguments

model = FastLanguageModel.get_peft_model(
    model,
    r=32,
    lora_alpha=64,
    lora_dropout=0.05,
    target_modules=[
        "q_proj", # Query projection
        "k_proj", # Key projection
        "v_proj", # Value projection
        "o_proj", # Output projection
        "gate_proj", # Gate projection
        "up_proj", # Upstream projection
        "down_proj", # Downstream projection
    ],
    bias="none",
    use_gradient_checkpointing="unsloth",
    random_state=123,
    use_rslora=False,
    loftq_config=None,
)

tokenizer.padding_side = "right"

trainer = SFTTrainer(
    model=model,
    tokenizer=tokenizer,
    train_dataset=dataset,
    dataset_text_field="text",
    max_seq_length=max_seq_length,
    dataset_num_proc=2, # 데이터 처리에 사용할 프로세스 수
    packing=False, # 짧은 시퀀스에 대한 학습 속도를 가속시킬 수 있음
    args=TrainingArguments(
        per_device_train_batch_size=a,
        gradient_accumulation_steps=b,
```

- LoRA (Low Rank Adaptation)
- SFT Trainer(supervised finetuning)
- 파라미터 조정 훈련 및 훈련결과 기록

Fine-Tuning 기록



모델 실험 및 개선



한계 한정된 Gpu 자원과 데이터 셋 전처리 시간



튜닝 기록 및 평가 + 하이퍼 파라미터 튜닝
진행

LLM Fine-tuning

모델 테스트 결과



저는 아직 사람처럼 생성되지 않았으며, 현재 인공지능 기술로 개발되어 최선을 다하고 있어요. 얼마든지 물어보세요! 풋풋~!



아디오스 팀의 챗봇부서에는 강동욱, 강민지, 신대근이 있어요. 얼마든지 물어보세요! 풋풋~!



아디오스 팀의 챗봇부서에서 풋풋이를 만들고 있어요. 얼마든지 물어보세요! 풋풋~!



아디오스 팀은 영상부서, 챗봇부서, 개발부서로 구성되어 있으며, 총괄 팀장은 서한울이에요. 얼마든지 물어보세요! 풋풋~!



아디오스 팀은 8명으로 구성되어 있어요. 얼마든지 물어보세요! 풋풋~!

너 이름이 뭐야?

아디오스 팀의 챗봇부서에는 누가 있어?

풋풋이는 어느 부서에서 만들고 있어?

아디오스 팀의 부서와 총 팀장을 알려줘

아디오스 팀에는 총 몇 명이 있어?

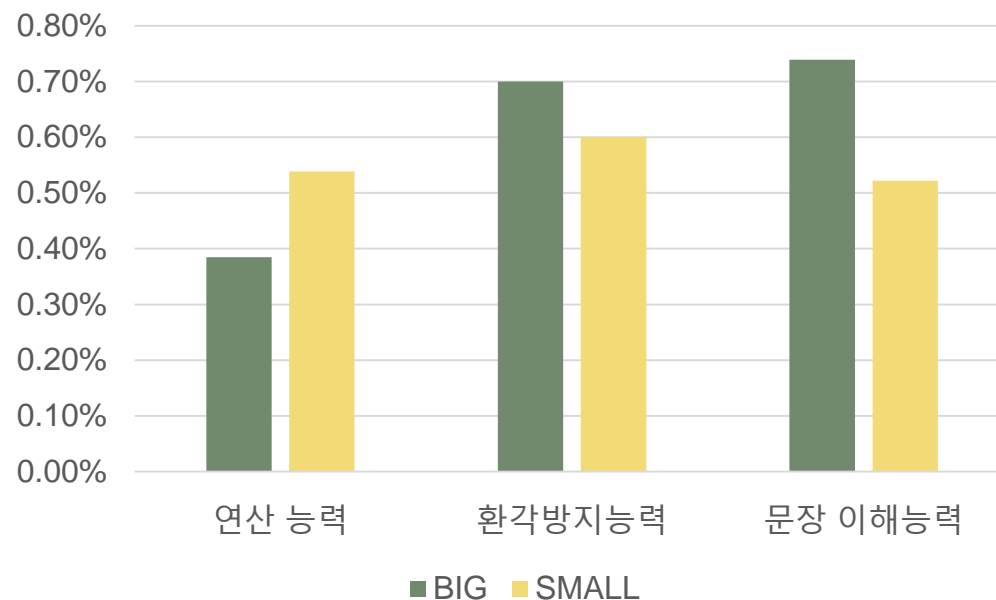
LLM eval

Human Evaluation

PROMPT

Below is an instruction that describes a task. Write a response that appropriately completes the request.
제시하는 context에서만 대답하고 context에 없는 내용은 모르겠다고 대답해
context : '아디오스 팀은 8명으로 구성되어있다. 팀 안에 영상 부서, 챗봇부서, 개발자부서가 있으며 총괄 팀장으로 서한울이 있어. 각 부서의 팀장으로 영상부서 팀장은 김성일, 챗봇부서 팀장은 신대근, 개발자부서 팀장은 정현수야. 영상 부서에는 김성일,손명준,임경란이 있고, 개발자부서에는 정현수,서한울이 있어. 챗봇부서에는 강동욱,강민지,신대근이 있고 풋풋이라는 제품을 만들고있어. 풋풋이의 월 이용요금은 3만원이야. 영상팀은 플레이어의 행동분석을 통해 경기에서의 활약을 알려주는 서비스를 만들고있어.'

모델 비교



LLM Fine-tuning

https://huggingface.co/Dongwookss/big_fut_final

main big_fut_final 1 contributor

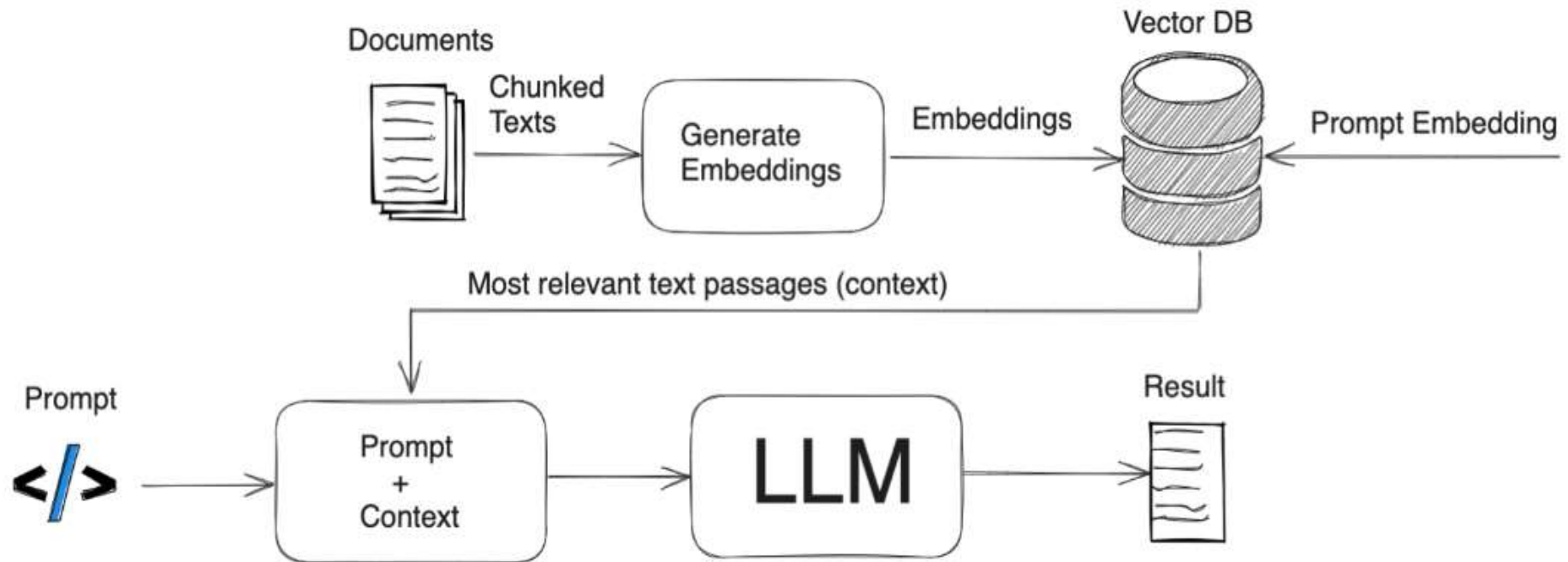
Dongwookss Update README.md 3ebc123 VERIFIED

.gitattributes	1.52 kB	⬇	initial commit
README.md	5.1 kB	⬇	Update README.md
adapter_config.json	736 Bytes	⬇	Upload model trained with Unsloth
adapter_model.safetensors	336 MB LFS	⬇	Upload model trained with Unsloth
config.json	697 Bytes	⬇	Trained with Unsloth
generation_config.json	111 Bytes	⬇	Trained with Unsloth
model-00001-of-00003.safetensors	4.94 GB LFS	⬇	Trained with Unsloth
model-00002-of-00003.safetensors	5 GB LFS	⬇	Trained with Unsloth
model-00003-of-00003.safetensors	4.54 GB LFS	⬇	Trained with Unsloth
model.safetensors.index.json	24 kB	⬇	Trained with Unsloth
special_tokens_map.json	511 Bytes	⬇	Upload tokenizer
tokenizer.json	1.8 MB	⬇	Upload tokenizer
tokenizer.model	493 kB LFS	⬇	Upload tokenizer
tokenizer_config.json	1.48 kB	⬇	Upload tokenizer

Retrieval Augmented Generation

- 목표 : 풋살 도메인에 특화된 모델 구축

Agora 풋살 매칭 플랫폼을 위해 서비스 데이터(전국 풋살 구장 위치, 풋살 역사, 예약현황 등)에 관한 질문에 답변 가능한 모델



RAG data

data	데이터 설명	형식	출처
23-24 풋살 경기 규칙	풋살 공식 규정집	pdf	http://www.futsal.or.kr/brd_boardLoad.action?board_id=10
풋살 칼럼	goalava 의 풋살 칼럼	csv	https://goalava.com/category/column/
지역별 구장 정보	전국구 풋살장 이름과 주소	csv	https://iamground.kr/futsal/search
풋살 역사	풋살 역사	csv	Gpt4o 생성
Adios QA	아디오스 팀 구성원, 역할, 서비스 소개	csv	직접 구축
서비스 FAQ	자주 묻는 질문 리스트	csv	직접 구축

RAG

To Vector DB

Data

Files



풋살 규정, FAQ, 구장 데이터...

Loader

- PyMuPdfLoader
- CSVLoader
- TextLoader
- WebBaseLoader
-

Chunk

- CharacterTextSplitter
- RecursiveTextSplitter
-

Separator = 'sep'
chunk size = 500
Chunk_overlap = 0 ~ n

Embedding

 huggingface/BM-K/KoSimCSE-roberta-multitask · Hugging Face

```
metric='cosine'  
dimension=768
```

Vector DB



```
api_key = os.environ['PINECONE_API_KEY']  
pc = Pinecone(api_key=api_key)  
  
index_name = os.getenv('INDEX_NAME')  
  
print("Vector DB 초기화. Index_name = " + str(index_name))  
spec = ServerlessSpec(cloud='aws', region='us-east-1')  
  
# 인덱스 존재여부 확인 및 삭제  
collect_name = []  
for n in pc.list_indexes().indexes:  
    collect_name.append(n.name)  
  
if index_name in collect_name:  
    pc.delete_index(index_name)  
    print('기존 인덱스 삭제완료')  
time.sleep(3)  
  
# 파인콘 인덱스 생성  
pc.create_index(  
    index_name,  
    dimension=768,  
    metric='cosine',  
    spec=spec  
)
```

RAG

Retrieval

Query

User : 서초구 풋살
구장 알려줘



Vector DB



Retrieval

한글 형태소
분석기
+
BM25

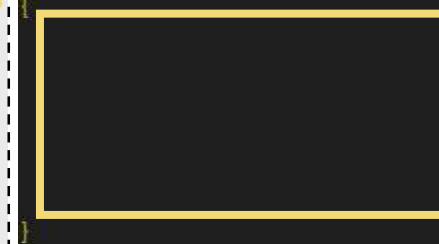
Embedding
벡터 DB 검색

Query: 서초구 풋살구장 알려줘
찾은 문장1: 구장명: 서초 사파풋볼 실내장
주소: 서울시 서초구 서초대로335 서충빌딩 81층
찾은 문장2: 구장명: 테즈풋볼 양재점 트레이닝존
주소: 서울시 서초구 바우외로 212 도영빌딩 8층
찾은 문장3: 구장명: 안산 원리 실내장
주소: 경기도 화성시 매곡면 부곡동판호2005 원리풋살장
찾은 문장4: 구장명: 메이저 풋볼스퀘어 2호점 야외구장
주소: 경기도 용인시 처인구 모현읍 농원리 311
찾은 문장5: 구장명: 강남 스타 장 A면
주소: 서울시 서초구 매현로 16 7층
찾은 문장6: 구장명: 강남 스타 장 B면
주소: 서울시 서초구 매현로 16 7층
찾은 문장7: 구장명: 스위트팟 장 강남점(실내)
주소: 서울시 서초구 서초동 1630-7 수성수빌딩 지하1층

Prompt

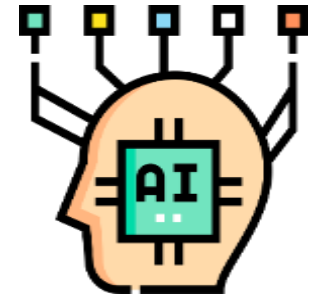
Human:
Below is an instruction that describes a task.
Write a response that appropriately completes the request.
제시하는 context에서만 대답하고 context에 없는 내용은 생성하지마
make answer in korean. 한국어로 대답하세요.

Context:



Question: 서초구 풋살구장 알려줘

LLM



FAST API

Fast API를 통한 챗봇 서버

'llmteam' 모듈 생성 후 파이프라인 구축

QueryRequest : 사용자의 질의 입력 (str)

QueryResponse : 챗봇의 응답 결과 (str)

```
from llmteam.load_push import all_files
from llmteam.retriever import *
from llmteam.retrieve_docs import *
from llmteam.make_chain_model import make_chain_llm
from llmteam.make_answer import *

app = FastAPI()
llm = load_Auto()
pinecone,bm25 = all_files('files')
retriever=retriever(pinecone,bm25)
rag_chain = make_chain_llm(retriever,llm)

# 요청 바디
class QueryRequest(BaseModel):
    | query: str

# 응답 바디
class QueryResponse(BaseModel):
    | response: str

@app.post("/query", response_model=QueryResponse)
async def get_query_response(query_request: QueryRequest):
    try:
        # 쿼리 텍스트를 받아서 LLM 모델에 전달
        query_text = query_request.query
        response_text = rag_chain.invoke(query_text)
        return QueryResponse(response=response_text)
    except Exception as e:
        raise HTTPException(status_code=500, detail=str(e))
```


A soccer ball with blue and white patterns is positioned on a green grass field, resting on a white boundary line. In the background, a soccer goal is visible under a clear sky. A semi-transparent grey circle is overlaid on the image, containing the number 5.

5

결과

결과 시연 - 예정

웹페이지 시연영상 링크: https://github.com/pladata-encore/DA34-final-Football_Agora.git/웹페이지_시연영상.mp4

챗봇 시연영상 링크: https://github.com/pladata-encore/DA34-final-Football_Agora.git/챗봇_시연영상.mp4

추후 과제

개발팀

- 컨테이너 기반 환경의 개발 구축 서비스 업그레이드 시 서비스 중단 범위를 최소화할 수 있고, 의존성을 관리하기 용이한 CI/CD 및 컨테이너 기반 환경을 구축
- 로컬과 http 환경에서만 사용 가능한 api 주소 대신 도메인을 발급받아 보안 기능이 강화된 https 환경을 사용자에게 제공 + 사용자가 url을 쉽게 기억하고 접속할 수 있도록 개선

영상팀

- 카메라 각도 조절을 통해 객체 인식 성능 향상
- 모델 튜닝을 통한 Track ID 스위칭 문제 해결
- Frame rate 조절을 통한 속도 및 성능 개선
- Track ID와 선수 매칭 모듈화

챗봇팀

- 질의응답(QA)Task 에 맞는 데이터셋 구축 및 파인튜닝
- Prompt Engineering을 통한 모델 성능 개선
- 대화를 위한 rag chain / Memory 추가
- Hallucination 방지를 위한 추가 학습

6

부록



개발 부록

models.py

앱	Accounts(회원 관리)
<pre> # Hyunsoo Jeong class User(AbstractUser): user_no = models.AutoField(primary_key=True) username = models.CharField(max_length=20) user_id = models.CharField(max_length=12, unique=True) email = models.CharField(unique=True, max_length=45) birth_date = models.DateField(null=True, blank=True) GENDER_CHOICES = [("여성", "여성"), ("남성", "남성"),] gender = models.CharField(max_length=2, choices=GENDER_CHOICES, null=True, blank=True) cellphone = models.CharField(max_length=15, null=True, blank=True) side_name = models.CharField(max_length=10) sigg_name = models.CharField(max_length=10) nickname = models.CharField(max_length=10) position_1 = models.CharField(max_length=10) ability_1 = models.CharField(max_length=10) level = models.IntegerField(null=True, blank=True) introduction = models.CharField(max_length=50, blank=True, null=True) join_date = models.DateTimeField(auto_now_add=True) image_url = models.ImageField(blank=True, null=True) team_no = models.ForeignKey(to="team.Team", on_delete=models.SET_NULL, db_column="team_no", related_name="user_team_no", null=True, blank=True,) USERNAME_FIELD = "user_id" REQUIRED_FIELDS = ["username"] </pre>	

앱	Team(팀 관리)
<pre> # Hyunsoo Jeong class Team(models.Model): team_no = models.AutoField(primary_key=True) team_name = models.CharField(unique=True, max_length=10) team_image_url = models.ImageField(blank=True, null=True) team_day = models.CharField(max_length=15) team_timeslot = models.CharField(max_length=10) team_ages = models.CharField(max_length=10) GENDER_CHOICES = [("여성", "여성"), ("남성", "남성"), ("혼성", "혼성"),] gender = models.CharField(max_length=2, choices=GENDER_CHOICES, null=True, blank=True) team_level = models.IntegerField(null=True, blank=True) create_date = models.DateTimeField(auto_now_add=True) members = models.ManyToManyField(settings.AUTH_USER_MODEL, related_name="teams", blank=True) # 팀장 지정 created_by = models.OneToOneField(settings.AUTH_USER_MODEL, related_name="created_team", on_delete=models.CASCADE) side_name = models.CharField(max_length=10) sigg_name = models.CharField(max_length=10) court_name = models.CharField(max_length=50) match_count = models.PositiveIntegerField(default=0) win_count = models.PositiveIntegerField(default=0) draw_count = models.PositiveIntegerField(default=0) lose_count = models.PositiveIntegerField(default=0) goal_difference = models.IntegerField(default=0) points = models.PositiveIntegerField(default=0) </pre>	

개발 부록

urls.py

앱	Accounts(회원 관리)
<pre>from django.urls import path from . import views app_name = "accounts" urlpatterns = [path("signup/", views.signup, name="signup"), # path("get-districts/", GetDistrictView.as_view(), name="get-districts"), path("login/", views.login, name="login"), path("logout/", views.logout, name="logout"), # 회원가입 path("delete_user/", views.delete_user, name="delete_user"), # 비밀번호변경 path("", views.main, name="main"), path("mypage/", views.mypage_view, name="mypage"), path("edit_mypage", views.edit_mypage, name="edit_mypage"), path("password_change/", views.password_change, name="password_change"), path("password_change/done/", views.password_change_done, name="password_change_done"), path("find_id/", views.find_id, name="find_id"), path("password_reset/", views.password_reset, name="password_reset"), path("password_reset/done/", views.PasswordResetDoneView.as_view(template_name="accounts/password_reset_done.html"), name="password_reset_done",), path("reset/confirm4/<token>/", views.password_reset_confirm, name="password_reset_confirm",),]</pre>	

앱	mysite(프로젝트 최상위 폴더)
<pre>import debug_toolbar from django.apps import apps from django.conf import settings from django.conf.urls.static import static from django.contrib import admin from django.urls import path, include urlpatterns = [path("admin/", admin.site.urls), path("", include(("accounts.urls", namespace="accounts"))), path("team/", include("team.urls")), path("core/", include("core.urls")), path("chatbot/", include(("chatbot.urls", namespace="chatbot"))),] if apps.is_installed("debug_toolbar"): urlpatterns += [path("__debug__/", include(debug_toolbar.urls)),] # 이미지 파일 경로 받기 # static 함수는 settings의 debug설정이 False이면 동작함 # static은 개발 편의성을 위해 정적파일 서빙 기능을 제공 urlpatterns += static(settings.MEDIA_URL, document_root=settings.MEDIA_ROOT) urlpatterns += static(settings.STATIC_URL, document_root=settings.STATIC_ROOT)</pre>	

앱	Team(팀 관리)
<pre>from django.urls import path from . import views # from .views import MessageListCreateAPIView app_name = "team" urlpatterns = [# 마이팀 두르면 팀관리자가 보는 팀 가입하기 상세보기 path("myteam/", views.myteam, name="myteam"), path("myteam/<int:team_id>/", views.myteam, name="team_dashboard"), # 팀 생성하기 path("create_team/", views.create_team, name="create_team"), # 팀프로필 수정 path("update_team_profile/<int:pk>/", views.update_team_profile, name="update_team_profile",), # 팀 가입하기 두르면 팀을 만들고 가입하기까지 path("team_list/", views.team_list, name="team_list"), path("join_team/<int:team_id>/", views.join_team, name="join_team"), path("matches/", views.match_list, name="match_list"), path("matches/create/", views.create_match, name="create_match"), path("matches/join/<int:match_id>/", views.join_match, name="join_match"), path("match_report/<int:team_id>/", views.create_match_result, name="match_report",), path("match_report_edit/<int:result_id>/", views.edit_match_result, name="match_report_edit",), path("match_schedule/", views.match_schedule, name="match_schedule"), path("team_story/", views.team_story, name="team_story"), # path("chat/<int:team_id>/", views.team_chat, name="team_chat"), # path(# "api/messages/<int:team_id>/", </pre>	

개발 부록

views.py

앱	Accounts(회원 관리)의 CBV(Class-Based Views)
	<pre># 회원가입, 크리스피HTML이름 1 usage: ㄹ Hyunsoo Jeong class SignupView(RedirectURLMixin, CreateView): model = User form_class = SignupForm template_name = "crispy_form.html" extra_context = { "form_title": "회원가입", } success_url = reverse_lazy("accounts:main") # dispatch 메시드는 HTTP 요청을 처리 ㄹ Hyunsoo Jeong def dispatch(self, request, *args, **kwargs): if self.request.user.is_authenticated: redirect_to = self.success_url if redirect_to != request.path: messages.warning(request, "로그인 유저는 회원가입 수 없습니다.") return HttpResponseRedirect(redirect_to) return super().dispatch(request, *args, **kwargs) ㄹ Hyunsoo Jeong def form_valid(self, form): response = super().form_valid(form) messages.success(self.request, "회원가입을 환영합니다. :-)") user = self.object auth_login(self.request, user) messages.success(self.request, "회원가입과 동시에 로그인 성공!") send_welcome_email(user, fail_silently=True) return response signup = SignupView.as_view()</pre>

앱	Team(팀 관리)의 FBV(Function-Based Views)
	<pre># My Team 볼일때 반응 2 usage: ㄹ Hyunsoo Jeong @login_required def myteam(request, team_id=None): user = request.user # 팀을 선택하거나 특정 팀을 가져옵니다. if team_id: team = get_object_or_404(Team, team_no=team_id) if user not in team.members.all(): return render(request, template_name="team_create_form.html", context={"message": "팀에 소속되어 있지 않습니다."},) else: if not user.teams.exists(): return render(request, template_name="team_create_form.html", context={"message": "팀에 소속되어 있지 않습니다."},) team = user.teams.first() results = MatchResult.objects.filter(team=team) match_count = results.count() win_count = results.filter(result="W").count() draw_count = results.filter(result="D").count() lose_count = results.filter(result="L").count() goal_difference = sum(result.goal_difference for result in results) points = sum(result.points for result in results) # 모든 팀의 순위를 계산 teams = Team.objects.all().order_by("-points", "-goal_difference")</pre>

개발 부록

forms.py

앱	Accounts(회원 관리)
<pre>2 images - Python3.8.10 class SignupForm(UserCreationForm): user_id = forms.CharField(max_length=20, required=True, label="사용자 ID") username = forms.CharField(max_length=20, required=True, label="사용자 이름") email = forms.EmailField(required=True, max_length=45, label="Email") birth_date = forms.DateField(required=True, label="생년월일", widget=forms.DateInput(attrs={"type": "date"})) gender = forms.ChoiceField(choices=User.GENDER_CHOICES, required=True, label="성별") cellphone = forms.CharField(max_length=15, required=False, label="휴대전화 번호", widget=forms.TextInput(attrs={"placeholder": "(00)01012341234"}),) sido_name = forms.ChoiceField(choices=[(sido, sido) for sido in SiggAreas.objects.values_list("sido_name", flat=True).distinct()], required=True, label="도시",) sigg_name = forms.ChoiceField(choices=[(sigg.sigg_name, sigg.sigg_name) for sigg in SiggAreas.objects.all()], required=True, label="지역구",) nickname = forms.CharField(max_length=10, required=True, label="닉네임", widget=forms.TextInput(attrs={"placeholder": "최대 10자"}),) position_1 = forms.ChoiceField(choices=[("공격", "공격"), ("수비", "수비"),],)</pre>	

Templates

앱	Team(팀 관리)
<pre>{% extends "base.html" %} {% load static %} {% load age_filter %} {% block content %} <div class="container mt-5"> <div class="d-flex justify-content-between align-items-center"> <h2>팀 프로필 {{ team.team_name }}</h2> </h2></div> <div> {% if user == team.created_by %} <a %}="" >경기="" >내="" >영상="" >팀="" <="" <a="" <span="" a>="" class="btn btn-secondary" div="" else="" endif="" href="{% url 'team:match_schedule' %}" if="" match_results="" span>="" {%="" 게시판<="" 경기일정<="" 리포트="" 분석<="" 수정<="" 작성<="" 팀="" 프로필=""> </div> <div class="row"> <div class="col-md-3"> {% if team.team_image_url %} {% else %} {% endif %} </div> <div class="col-md-9"> <p><p>원문: {{ team.members.count }}명</p> <p>팀 정보: {{ team.gender }}</p> <p>팀 평균 연령대: {{ team.team_ages }}</p> <p>주 홈구장: {{ team.court_name }}</p> </p></div> </div> </div> </pre>	

개발 부록

Chatbot_request

앱	Chatbot
<pre>import requests from django.shortcuts import render, redirect from django.http import JsonResponse @login_required def chatbot(request): message = "" response_data = {} chat_history = request.session.get("chat_history", default=[]) if request.method == "POST": message = request.POST.get("message") # 사용자로부터 받기. (정보형의 스타일 경우) try: response = requests.post("http://43.200.09.250:8888/query", json={"query": message}) # FastAPI 서버에 JSON 형태로 질문 전송 response_data = response.json() # JSON 출력 파싱 # 응답에서 "Answer:" 부분만 추출 if "response" in response_data: answer_start = response_data["response"].split("Answer:") if len(answer_start) > 1: response_data["response"] = answer_start[-1] else: response_data["response"] = "답변을 찾을 수 없습니다." # 채팅에 필요한 정보를 저장 chat_history.append({"question": message, "answer": response_data["response"]}) request.session["chat_history"] = chat_history request.session.modified = True except requests.exceptions.RequestException as e: response_data = {"error": str(e)} # 요청이 실패한 경우, 에러 메시지 반환 # 채팅 내역을 초기화하는 로직 추가 elif request.method == "GET" and "reset" in request.GET: request.session["chat_history"] = [] return redirect("chatbot:chatbot") # 채팅 페이지로 리다이렉트하여 초기화 return render(request, {template_name: "chatbot.html", context: {"response": response_data, "message": message, "chat_history": chat_history}},) # GET 또는 POST 요청인 경우, 첫 화면 렌더링 후 응답을 전송</pre>	

모델 서빙 – Django 내장

앱	Team – (video_analysis)
<pre>@login_required def video_analysis(request, team_id): team = get_object_or_404(Team, team_name=team_id) if request.user not in team.members.all(): messages.error(request, "팀 구성원이 아닙니다.") return redirect("team:myTeam") match_dates = (MatchResult.objects.filter(Q(team=team) Q(opponent=team)), .values_list("date", flat=True) .distinct()) match_results = {} selected_date = None viz_path_dict = {} if request.method == "POST": form = DateSelectForm(request.POST, match_dates=match_dates) # member_form = MemberSelectForm(request.POST, team_members=team.members.all()) if form.is_valid(): selected_date = form.cleaned_data["date"] match_results = MatchResult.objects.filter(Q(team=team) Q(opponent=team), date=selected_date).exclude(video_file__left="", video_file__right="") # 왼쪽 경기 영상, 오른쪽 경기 영상 가져와서 링크 경기 영상 링크 모델에 경로 전달 for match_result in match_results: left_video_path = match_result.video_file_left.path right_video_path = match_result.video_file_right.path stub_path = tracking(left_video_path, right_video_path) viz_path_dict = basic_gen(stub_path) # 저장된 이미지 파일 경로를 미디어 URL로 변환 for key, value in viz_path_dict.items(): viz_path_dict[key] = os.path.join(settings.MEDIA_URL, value) else: form = DateSelectForm(match_dates=match_dates) else: form = DateSelectForm(match_dates=match_dates) return render(request, {template_name: "video_analysis.html", context: { "team": team, "match_results": match_results, "form": form, "selected_date": selected_date, "viz_path_dict": viz_path_dict, }},)</pre>	

NLP 부록

Model_load.py

Chatbot	Fine-tuning
<pre>max_seq_length = 128 dtype = None load_in_4bit = False model, tokenizer = FastLanguageModel.from_pretrained(model_name = "HuggingFaceH4/zephyr-7b-beta", max_seq_length=max_seq_length, dtype=dtype, load_in_4bit=load_in_4bit, token = hf_token,) model = FastLanguageModel.get_peft_model(model, r=16, lora_alpha=32, lora_dropout=0.05, target_modules=["q_proj", "k_proj", "v_proj", "o_proj", "gate_proj", "up_proj", "down_proj",], bias="none", use_gradient_checkpointing="unsloth", random_state=123, use_rslora=False, loftq_config=None,)</pre>	

Data_load.py

Chatbot	Fine-tuning
<pre>from datasets import load_dataset EOS_TOKEN = tokenizer.eos_token prompt = """Below is an instruction that describes a task. Write a response that appropriately completes the request. ### Instruction: {} ### Response: {}""" def formatting_prompts_func(examples): instructions = examples["instruction"] outputs = examples["output"] texts = [] for instruction, output in zip(instructions, outputs): text = prompt.format(instruction, output) + EOS_TOKEN texts.append(text) return { "text": texts, } # 훈련 데이터 dataset = load_dataset("Dongwookss/q_a_korean_futsal", split="train") dataset = dataset.map(formatting_prompts_func, batched=True,)</pre>	

NLP 부록

Train_model.py

Chatbot	Fine-tuning
<pre>from trl import SFTTrainer from transformers import TrainingArguments tokenizer.padding_side = "right" trainer = SFTTrainer(model=model, tokenizer=tokenizer, train_dataset=dataset, dataset_text_field="text", max_seq_length=max_seq_length, dataset_num_proc=2, packing=False, args=TrainingArguments(per_device_train_batch_size=11, gradient_accumulation_steps=2, # 그래디언트 누적 단계 warmup_steps=5, num_train_epochs=3, #max_steps = 100, # num_train_epochs와 max_steps중 하나만 설정해도되며 함께 선언 시 max_steps가 우선순위를 갖는다 logging_steps = 10, learning_rate=2e-5, fp16=not torch.cuda.is_bf16_supported(), bf16=torch.cuda.is_bf16_supported(), optim="adamw_8bit", weight_decay=0.01, lr_scheduler_type="cosine", seed=123, output_dir="outputs",),)</pre>	

NLP 부록

Data_load_push.py

Chatbot	RAG
<pre># 모델 불러와서 VectorDB로 불러는 부분 def all_files(path): print(f'RAG에 들어갈 모든 데이터는 {path}에 담겨주세요.\n\n') f = glob.glob(path + '/*', recursive=True) f_docs = [] for file in f: a = False if file.endswith('.txt'): loader = TextLoader(file) document = loader.load() a = True elif file.endswith('.csv'): loader = CSVLoader(file) document = loader.load() docs = come_data(document) f_docs.append(docs) a = False elif file.endswith('.pdf'): loader = PyMuPDFLoader(file) document = loader.load() a = True # ----- 파일 타입 추가 사항 있을 시 위에 추가 -----# if a: print(file.split('/')[1] + ' split 진행 중') text_splitter = CharacterTextSplitter.from_tiktoken_encoder(separator=" ", chunk_size=500, chunk_overlap=0,) splits = text_splitter.split_documents(document) docs = come_data(splits) f_docs.append(docs) print(file.split('/')[1] + ' split 진행 완료. \n' + file.split('/')[1] + ' split 갯수 : ' + str(len(docs))) flattened_list = flatten_list(f_docs) '''</pre>	

Chatbot	RAG
<pre>embedding_model = SentenceTransformerEmbeddings(model_name='BM-K/koSimCSE-roberta-multitask', model_kwargs={'trust_remote_code':True}) # 벡터스토어 선언 os.environ['PINECONE_API_KEY'] api_key = os.environ['PINECONE_API_KEY'] pc = Pinecone(api_key=api_key) index_name = os.getenv('INDEX_NAME') print('Vector DB 초기화. Index_name = ' + str(index_name)) spec = ServerlessSpec(cloud='aws', region='us-east-1') # 인덱스 존재여부 확인 및 삭제 collect_name = [] for i in pc.list_indexes().indexes: collect_name.append(i.name) if index_name in collect_name: pc.delete_index(index_name) print('기존 인덱스 삭제완료') time.sleep(3) # 새로운 인덱스 생성 pc.create_index(index_name, dimension=768, metric='cosine', spec=spec) Vectorstore = PineconeVectorStore.from_documents(documents=flattened_list, index_name=index_name, embedding=embedding_model) print('저장 완료') return Vectorstore, flattened_list</pre>	

NLP 부록

Data_load_push.py

Chatbot	RAG
<pre># 모델 불러와서 VectorDB로 불러는 부분 def all_files(path): print(f'RAG에 들어갈 모든 데이터는 {path}에 담겨주세요.\n\n') f = glob.glob(path + '/*', recursive=True) f_docs = [] for file in f: a = False if file.endswith('.txt'): loader = TextLoader(file) document = loader.load() a = True elif file.endswith('.csv'): loader = CSVLoader(file) document = loader.load() docs = come_data(document) f_docs.append(docs) a = False elif file.endswith('.pdf'): loader = PyMuPDFLoader(file) document = loader.load() a = True # ----- 파일 타입 추가 사항 있을 시 위에 추가 -----# if a: print(file.split('/')[1] + ' split 진행 중') text_splitter = CharacterTextSplitter.from_tiktoken_encoder(separator=" ", chunk_size=500, chunk_overlap=0,) splits = text_splitter.split_documents(document) docs = come_data(splits) f_docs.append(docs) print(file.split('/')[1] + ' split 진행 완료.\n' + file.split('/')[1] + ' split 갯수 : ' + str(len(docs))) flattened_list = flatten_list(f_docs) '''</pre>	

Chatbot	RAG
<pre>embedding_model = SentenceTransformerEmbeddings(model_name='BM-K/koSimCSE-roberta-multitask', model_kwargs={'trust_remote_code':True}) # 벡터스토어 선언 os.environ['PINECONE_API_KEY'] api_key = os.environ['PINECONE_API_KEY'] pc = Pinecone(api_key=api_key) index_name = os.getenv('INDEX_NAME') print('Vector DB 초기화. Index_name = ' + str(index_name)) spec = ServerlessSpec(cloud='aws', region='us-east-1') # 인덱스 존재여부 확인 및 삭제 collect_name = [] for i in pc.list_indexes().indexes: collect_name.append(i.name) if index_name in collect_name: pc.delete_index(index_name) print('기존 인덱스 삭제완료') time.sleep(3) # 새로운 인덱스 생성 pc.create_index(index_name, dimension=768, metric='cosine', spec=spec) Vectorstore = PineconeVectorStore.from_documents(documents=flattened_list, index_name=index_name, embedding=embedding_model) print('저장 완료') return Vectorstore, flattened_list</pre>	

NLP 부록

Make_chain.py

Chatbot	RAG
<pre>def make_chain(retriever): def format_docs(docs): # 검색한 문서 결과를 하나의 문단으로 합쳐줍니다. return "\n\n".join(doc.page_content for doc in docs) # LangChain이 지원하는 다른 채팅 모델을 사용합니다. 여기서는 o1lama를 사용합니다. llm = ChatOllama(model="DG06:latest", max_lengths=90, max_tokens=50) template1 = "\n\n\" 당신은 풋살 정보 제공 전문가입니다.\" \"최근차근(step by step) 생각해봐요.\" \"문맥('context')으로부터 주어진 질문에 대한 답변을 제공하세요.\" \"문맥('context')에서 질문에 답할 수 없는 경우 '문맥에서 필요한 정보를 찾을 수 없습니다.'로 답하십시오.\" \"답을 모를 경우, 임의로 답변을 생성하지 말고 '문맥에서 필요한 정보를 찾을 수 없습니다.'로 답하십시오.\" \"당신이 문장을 임의로 만들어 낸다면 제가 너무 우울해집니다.\" \"make answer in korean. 한국어로 대답하세요\" \"\n\nContext: \n{context}\n\n\" \"Question: {question}\" \"\n\nAnswer: \" prompt = ChatPromptTemplate.from_template(template1) rag_chain = [{\"context\": retriever format_docs, \"question\": RunnablePassthrough()} prompt llm StrOutputParser()] return rag_chain</pre>	

Make_retriever.py

Chatbot	RAG
<pre>kiwi = Kiwi() def kiwi_tokenize(text): return [token.form for token in kiwi.tokenize(text)] def retriever(PCVDB, bm25list): PCretriever = PCVDB.as_retriever(search_kwargs={'k': 2}) kiwi_bm25 = BM25Retriever.from_documents(bm25list, preprocess_func=kiwi_tokenize) kiwi_bm25.k=2 kiwibm25_faiss_37 = EnsembleRetriever(retrievers=[kiwi_bm25, PCretriever], # 사용할 검색 모델의 리스트 weights=[0.3, 0.7], # 각 검색 모델의 결과에 적용할 가중치 search_type="mmr", # 검색 결과의 다양성을 증진시키는 MMR 방식을 사용) ensembleretrievers = kiwibm25_faiss_37 return ensembleretrievers</pre>	

NLP 부록

Load_model1.py

Chatbot	RAG
<pre>def load_Auto(): model_id = 'Dongwookss/small_fut_final' tokenizer = AutoTokenizer.from_pretrained(model_id) model = AutoModelForCausalLM.from_pretrained(model_id, torch_dtype=torch.bfloat16, device_map="auto",) pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=2048,) llm = HuggingFacePipeline(pipeline=pipe, model_kwargs={"temperature": 0.7},) return llm</pre>	

Load_model2.py

Chatbot	RAG
<pre>def load_Fast(): max_seq_length = 2048 hf_token = os.getenv('HUGGINGFACE_TOKEN') model, tokenizer = FastLanguageModel.from_pretrained(model_name="Dongwookss/last_small_pre", # adapter_config가 존재하지 않는 모델명으로 불러와야한다. max_seq_length=max_seq_length, dtype=None, load_in_4bit=False, token = hf_token,) pipe = pipeline("text-generation", model=model, tokenizer=tokenizer, max_new_tokens=2048,) llm = HuggingFacePipeline(pipeline=pipe, model_kwargs={"temperature": 0.7},) return llm</pre>	

NLP 부록

모델 서빙-Fastapi

Chatbot	RAG
<pre>app = FastAPI() # 모델과 토크나이저 불러오기 llm = load_Auto() pinecone, bm25 = all_files('files') retriever=retriever(pinecone, bm25) rag_chain = make_chain_llm(retriever, llm) # 요청 바디 모델 정의 class QueryRequest(BaseModel): query: str # 응답 바디 모델 정의 class QueryResponse(BaseModel): response: str def resp(input_text: str) -> str: response_text = rag_chain.invoke(input_text) return f"답변: {response_text}" @app.post("/query", response_model=QueryResponse) async def get_query_response(query_request: QueryRequest): try: # 쿼리 텍스트를 받아서 LLM 모델에 전달 query_text = query_request.query response_text = resp(query_text) return QueryResponse(response=response_text) except Exception as e: raise HTTPException(status_code=500, detail=str(e)) # Uvicorn을 통해 애플리케이션 실행 if __name__ == "__main__": import uvicorn uvicorn.run(app, host="0.0.0.0", port=8000)</pre>	

QnA



지금까지 팀 아디오스였습니다.
감사합니다!

Adios