

Fashion Coordination Recommendation Project

"What's In Your Closet?" Wrapup Report

[Service Link](#)[GitHub Repository](#)

RecSys. Track 7조 유쾌한발상

이채원, 김소미, 서현덕, 백승주, 유종문

1. 프로젝트 개요

사람들은 자신의 옷장을 보면서 입을 옷이 없다며 불편함을 많이 느끼고 있다. 실제로 가진 옷이 많음에도 불구하고 이를 활용해서 마음에 드는 다양한 코디를 할 수 없어서 발생하는 문제인데, 우리 팀은 이런 문제점을 파악해서 옷이 적더라도 최대한 많은 코디에 섞어서 사용할 수 있는 사용자의 옷장을 만들고자 했다. 그렇게 결정된 프로젝트가 <What's In Your Closet?>이며, 요약하자면 사용자가 가지고 있는 옷을 바탕으로 이와 가장 어울리는 제품을 추천하는 서비스이다.

A. 활용 도구

UpStage 측에서 제공받은 V100 GPU를 각 로컬과 SSH로 연결하였고, IDE로는 VSCode를 사용하여 개발하였다. 협업을 위해서는 GitHub, 일정관리를 위한 Jira와 Confluence를 사용하였다. 그 이외로 Zoom, Slack을 통한 의사소통을 통해 협업을 원활히 하고자 했다.

B. 기대 효과

사용자가 가진 옷을 다양한 코디에 활용할 수 있게 함으로써, 더 조화로운 옷장을 구성할 수 있을 것이다.

2. 프로젝트 팀 구성 및 역할

공통	문제 정의, 목표 설정 및 일정 수립 (계획 세분화)
이채원	프론트엔드(Streamlit), 백엔드(Fast API), Product Serving, 데이터베이스 설계
유종문	데이터 수집(크롤링) 및 전처리, Modeling(LightGCN), UI개선
김소미	데이터 수집(크롤링) 및 전처리, DB 설계 및 관리, 백엔드(GCP, Fast API), UI개선
서현덕	데이터 수집(크롤링) 및 전처리, Modeling(Rule Based)
백승주	프론트엔드(Streamlit) 설계, 백엔드(GCP) 관리, Fast API 문서화, Product Serving

3. 프로젝트 수행 절차 및 방법

A. 문제정의 / 목표설정

유저가 현재 가진 옷의 정보를 입력했을 때, 그 옷과 가장 매칭이 잘 되는 옷과 코디의 모습을 보여주는 것
MLOps의 전체적인 Life Cycle을 경험하고 Serving 되었을 때, 사용자가 사용하기 쉽고 만족할 UI를 구현하는 것

B. 프로젝트 사전 기획

i. 프로젝트 일정 수립 (Jira 사용)

'22.05.16 ~ '22.05.19	MUSINSA 데이터 크롤링 진행
'22.05.19 ~ '22.05.27	데이터 전처리, Streamlit을 사용하여 UI 설계 및 개선
'22.05.19 ~ '22.06.03	1차 End to End 구현 (프로토타입)
'22.05.20 ~ '22.05.27	Streamlit의 검색 기능 구현, 검색 결과 개선

'22.05.21 ~ '22.06.03 Model 구현 (Rule Based, Light GCN), GCP, Fast API 적용
 '22.06.01 ~ '22.06.03 GCP MySQL DB 데이터 적재 및 지속적인 업데이트
 '22.06.06 ~ '22.06.08 UI 개선 및 서비스 완성, GCP Backend와 Frontend 서버 구축

ii. 역할 분담

Backend + Frontend : GCP, Fast API를 사용한 백엔드 구성, streamlit을 통한 프론트 설계와 DB 설계

데이터 수집 및 전처리, 모델링: Selenium을 사용한 데이터 수집과 검수, 전처리 및 모델링 진행

iii. GitHub 버전 관리 규칙 (Branch 및 Commit 메시지 관리)


Commit Message

[TYPE][Author] message(#issue number)
 TYPE: FEAT, FIX, DOCS, TEST, STYLE, ...

Git Branch

feature/develop, feature/model, ...
 hotfix/{#issue number}

C. 데이터 수집(크롤링)



Product Info 제품정보

브랜드: THEGNEVERTHAT TN2388

시즌: 2019 SS

소재(소재): 300% 면

가격: 338,000원

구매 후기: 338

배송 정보

배송지: 서울특별시 강남구

배송 방법: 택배

배송 비용: 5,000원

배송 기간: 2~3일

배송지 변경: C대리점

Price Info 가격정보

무신사 판매가: 128,000원

무신사 최저가: 117,120 ~ 128,000원

무신사 적립금: 최대 8,120원

무신사 적립하는 전 상품 무료배송입니다.

무신사 적립금: 0원

무신사 적립금: 0원

제품(아이템) 속성

제품 번호	제품명	대분류	중분류
브랜드	제품 일련번호	성별	계절
누적 판매량	조회수	좋아요 수	평점
가격	제품 URL	제품 이미지 URL	R, G, B
제품 태그	주 소비 연령대	주 소비 성별	제품 핏

코디 속성

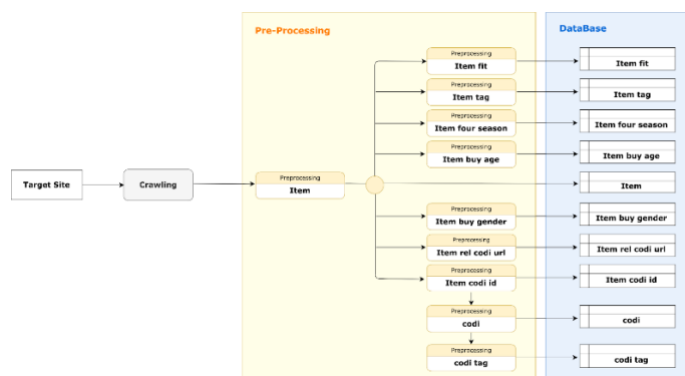
코디 번호	코디 URL	코디 이미지 URL	스타일
-------	--------	------------	-----

데이터 크롤링은 패션 사이트인 MUSINSA에서 진행하였다. MUSINSA에서 보이는 코디맵 또는 코디샷의 코디 정보들을 활용해서 아이템(옷)과 코디사이의 관계 또는 아이템과 아이템 사이의 관계를 파악하고자 했다. 그렇기에 각 아이템이나 코디에 포함된 정보(feature)들을 추출하여 저장하였다.

크롤링을 진행한 방법은 Python의 Selenium과 Chrome Driver를 사용해서 진행하였고, 이후에 크롤링 된 데이터들은 전처리 과정을 거쳐 Google Cloud Platform(GCP)의 DB에 적재하였다. 코디와 코디, 그리고 옷과의 관계/연관성을 파악하기 위해서 크롤링 전략으로 연관된 코디나 옷을 우선으로 크롤링을 진행했다. 예로, 하나의 코디 내에 여러 옷들이 존재하면 각각의 옷들과 연관된 다른 코디들을 다시 크롤링 했다.

D. 데이터 전처리

데이터 전처리는 크게 4가지로 진행하였다. (대분류 필터링, 결측치 처리, 색상 추출, 아이템 클러스터링)

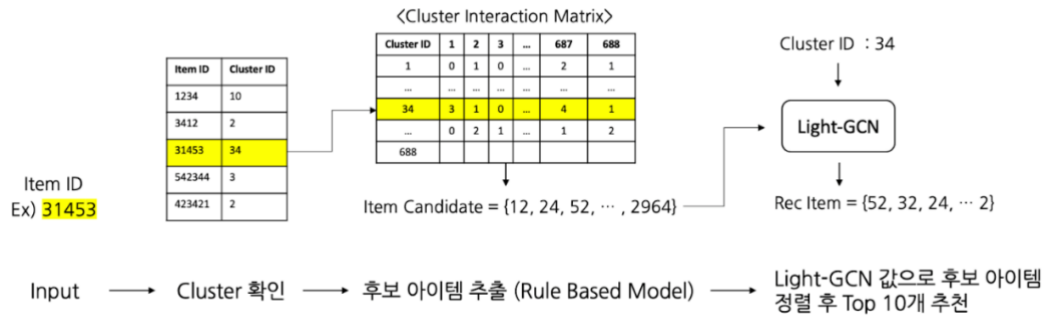


대분류 필터링은 우리의 문제 정의에 맞지 않는 옷의 대분류 (악세서리, 속옷 등)를 제거하는 과정을 진행하였고, 유

사한 대분류끼리 서로 통일해 주는 작업을 진행하였다. 결측치나 이상치 값에 대해서는 평균값이나 다른 속성의 정보를 사용해서 채워 넣었다. 예로, 아이템의 조회 수 결측치에 해당 아이템이 포함된 코디의 조회 수를 사용하였다.

색상을 추출하기 위해서 아이템의 이미지를 불러와서 'rembg'를 통해 배경을 제거하였고, Colorgram 라이브러리를 통해 현재 이미지에서 가장 많이 포함되어 있는 색상을 추출하도록 하였다. 잘못 라벨링 된 색상에 대해 검수하는 과정을 거치기도 하였다. 아이템 클러스터링은 어떤 두 아이템이 중분류와 색상이 동일하다면 같은 아이템으로 묶어주는 작업이다. 이를 통해 더 다양한 코디를 추천할 수 있는 계기가 되었다.

E. 모델 개요 (연산 처리 과정)

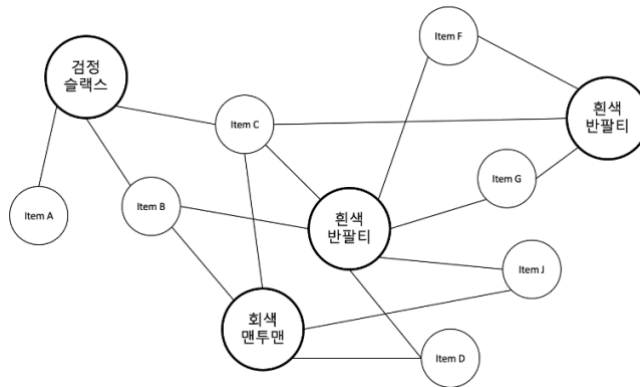


F. 모델 선정이유 및 분석

i. Rule Based Model

MUSINSA에서 크롤링 한 데이터는 상대적으로 sparsity가 높았다. 따라서, 머신러닝 혹은 딥러닝 모델을 적용하더라도 학습할 상호작용 데이터가 부족했기 때문에 좋은 성능을 내지 못했다. 결과적으로 우리 팀은 안정적인 성능을 보장하는 Rule-Based Model을 사용하기로 결정했다.

ii. Light-GCN



*Data Crawling Method

우리 데이터는 모든 아이템이 서로 끊어지지 않고 연결되어 있는 그래프 구조가 보장되어 있기 때문에, 이를 활용할 수 있는 그래프 계열 모델을 사용하는 것이 효과적이라고 생각했다.

*Data Sparsity

우리 데이터는 굉장히 sparse 했다. 이러한 sparsity를 보완하기 위해 직접 상호작용한 아이템뿐만 아니라, 연결 관계를 타고 들어가서 더 많은 정보를 고려하여 예측하는 그래프 계열 모델이 적합하다고 판단했다.

*Inference Speed

그래프 계열 모델 중에서 Light-GCN을 선택한 이유는, 모델 예측 속도가 가장 빠르기 때문이다. 모델이 결과를 산출하는 과정이 느리다면, 서비스 배포에 장애물이 될 수도 있기 때문에 상대적으로 빠른 모델을 선택했다.

G. 서비스 배포

i. Fast API

Streamlit에서 사용자가 웹페이지를 사용하며 요청하는 데이터베이스에 관련한 내용들에 Streamlit에서 직접 접근하는 것이 아닌 Fast API를 통해서 데이터베이스 서버에 접근했다. 이런 방식으로 Fast API를 사용해서 프론트 엔드, 백엔드, DB, 등의 프로세스를 연결했다.

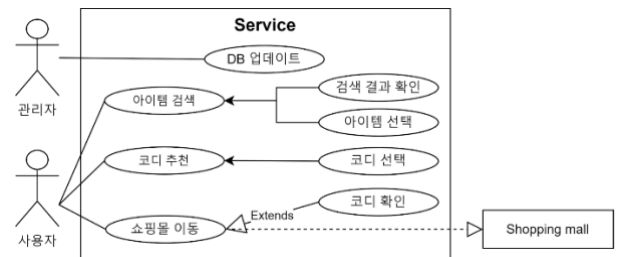
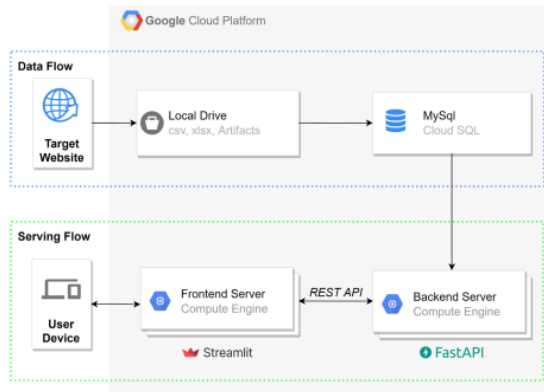
ii. Google Cloud Platform

데이터베이스 서버를 위해 SQL 서버를 만들고 DBeaver를 사용해 데이터베이스를 관리했다. Backend 서버와 Frontend 서버를 VM으로 만들어 따로 관리해서 완전히 모든 서비스를 로컬이 아닌 서버에서 사용할 수 있게 구축했다. 총 3개의 서버를 사용하였고, inference는 frontend 서버에서 진행했으며 학습은 로컬에서 진행했다.

iii. Streamlit

Streamlit으로 서비스를 빠르게 데모할 수 있는 웹 페이지를 구현할 수 있었다. session state를 활용했고, 페이지를 넘기는 것과 같은 효과를 주어 사용자가 서비스를 사용할 때 헛갈리지 않도록 했다. 또한, 설문의 피드백을 반영하여 사용자의 편의성을 개선했다.

H. 서비스 시스템 아키텍처 / Use case



I. 협업 과정

i. Git의 활용

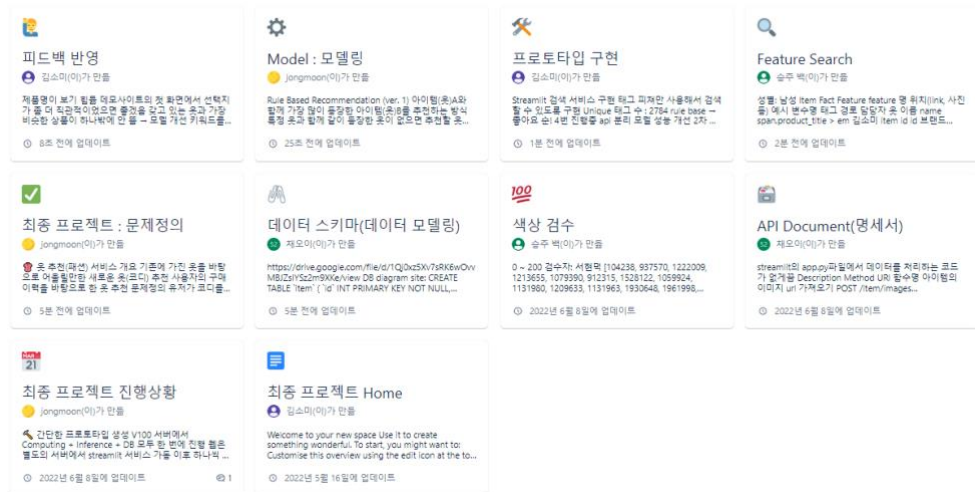
GitHub의 branch를 적극적으로 활용해서 각 모델 간의 실험을 독립적으로 설정하였고, 모든 팀원들이 실행한 코드들을 github를 통해 공유하며 협업을 할 수 있었다. 필요한 경우, pull request, merge conflict를 하여 더 빠른 코드 교환이 이루어질 수 있게 되었다.

ii. 일정관리를 위한 Jira

유형	키	요약	담당자	보고자	P	상태	책임	완료 날짜	현재까지	기타
📌	FINAL-41	백엔드 API 구현	👤 활동되지 않음	👤 김소희	🟡	완료됨	완료	2022년 5월 31일	2022년 6월 6일	...
📌	FINAL-40	GCP VM 만들기	👤 송주혁	👤 김소희	🟡	완료됨	완료	2022년 5월 31일	2022년 6월 6일	
📌	FINAL-38	DB에 데이터 올리기	👤 활동되지 않음	👤 김소희	🟡	완료됨	완료	2022년 5월 31일	2022년 5월 31일	
📌	FINAL-37	Data schema	👤 활동되지 않음	👤 채요이	🟡	완료됨	완료	2022년 5월 27일	2022년 5월 31일	
📌	FINAL-36	random 기능 구현	👤 채요이	👤 채요이	🟡	완료됨	완료	2022년 5월 27일	2022년 6월 6일	
📌	FINAL-35	GCP Database 연결	👤 활동되지 않음	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 5월 31일	
📌	FINAL-34	API 문서 추가화	👤 채요이	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 6월 6일	
📌	FINAL-32	사진 색 맞추	👤 활동되지 않음	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 6월 10일	
📌	FINAL-31	화그 그리기	👤 송주혁	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 6월 27일	
📌	FINAL-30	홈 버튼	👤 채요이	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 5월 27일	
📌	FINAL-29	사진 위치 옮기기	👤 송주혁	👤 채요이	🟡	완료됨	완료	2022년 5월 26일	2022년 5월 27일	

Jira의 로드맵과 보드를 활용하여 개인별 에픽과 하위 이슈들을 쉽게 정리하고 관리할 수 있었다. 로드맵의 경우 전체 일정이 bar 형태로 한눈에 보여 어떤 태스크가 진행이 안 되고 있는지 파악할 수 있었고, 팀원들이 현재 어떤 작업을 진행하고 있는지 실시간으로 확인 가능하였다.

iii. 다양한 실험정리, 인사이트 공유를 위한 Confluence Wiki



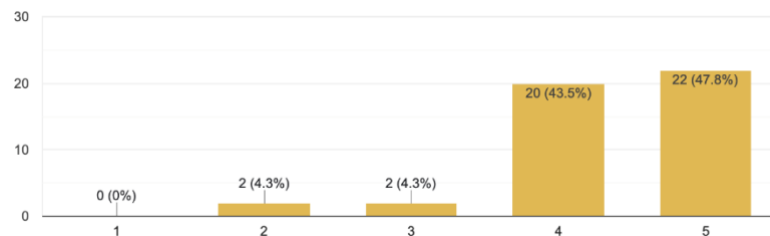
Confluence wiki를 통해서 각자 조사한 자료나 인사이트 또는 실험 결과들을 효과적으로 정리하고 공유할 수 있었다. Jira와 연동하여 일정에 맞게 진행사항을 확인할 수 있었고, 쉽게 정리하는 데 도움이 많이 된 플랫폼이다. 특히 최종 프로젝트에서는 현재 진행 상황을 실시간으로 공유하고 많은 의견을 나누는 것이 중요했는데, Confluence를 사용하여 각각의 진행 상황이나 피드백을 남길 수도 있었다.

4. 프로젝트 수행 결과

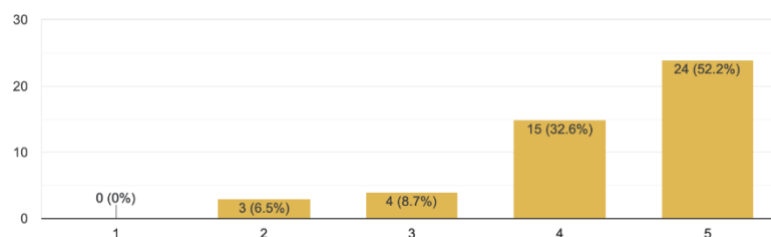
A. 서비스 평가 및 개선방향

서비스를 오픈하고 총 46명의 20, 30대를 바탕으로 서비스에 대한 피드백을 받은 결과

(1) 추천한 아이템이 어울린다고 생각하시나요? (4.32 / 5점)



(2) 추천된 코디가 마음에 드시나요? (4.28 / 5점)



(3) 피드백 결과를 바탕으로 하여 앞으로 개선할 수 있는 점들

- UI:** 더 높은 수준의 UI/UX를 위해서 Elastic Search와 같이 더 정교한 검색 툴 도입과 이에 따라 필요한 검색 태그의 개선이 필요하다.
- Data:** 아직 Sparsity를 완전하게 해결하지 못했기 때문에 더 많은 양의 데이터 크롤링이 필요하고, 더 폭넓은 아이템을 수집하기 위해서 다양한 플랫폼/소스에서 크롤링을 진행해야 한다.

- c. **Model:** Variational Auto Encoder 계열 모델을 사용하거나 Feature 정보를 사용하는 Content Based model을 사용하여 더 참신한 제품을 추천하는 모델을 도입이 필요하다.
- d. **MLOps:** 데이터 처리를 담당하는 Airflow와 프로세스를 관리하는 Kubernetes와 같은 툴을 도입하여 완전한 MLOps Life Cycle을 완성시킬 필요가 있다.

B. 시연결과

GitHub Repository에서 확인 가능: [What's-In-Your-Closet GitHub Repository](#)

5. 자체 평가 의견

좋았던 점

서비스 배포의 End-to-End를 경험한 것
끝까지 서비스를 개선하고자 노력한 것
새로운 기술을 사용하고자 노력한 것
팀원이 최종 결과에 모두 균등하게 기여한 것
GitHub 브랜치를 통해 코드와 결과를 빠르게 공유한 것

아쉬웠던 점

Airflow, Kubernetes를 사용하지 못한 것
더 다양한 모델을 사용하여 실험해 보지 못한 것
대용량의 트래픽을 발생시키는 실험을 해보지 못했던 것
MLOps 강의에 나온 많은 기법들을 다 적용해 보지 못한 것
Git Issue를 활발하게 사용하지 못했던 것
여러 사이트의 데이터를 크롤링 하여 사용하지 못한 것

1. 나는 내(팀) 학습목표를 달성하기 위해 무엇을 어떻게 했는가?

프로젝트를 효율적으로 관리하고 목표를 관리하기 위해서 Confluence를 가장 많이 활용했다. 진행 로그를 기록하고 다음 목표를 적어 다른 일을 하는 팀원들에게 공유했다. 그리고 전체적인 서비스 아키텍처를 그려서 공유함으로써 팀이 공동의 목표를 인식할 수 있게끔 했다. 이번엔 성능이 아닌 원하는 기능이 작동되는 웹 페이지 구현이 1순위 목표였기 때문에, 하나의 프로토타입을 기준으로 조금씩 발전시켜 나갔다. 또한, 많은 작업이 필요하기에 데이터 수집팀과 Frontend 팀으로 나누어 작업해서 빠르게 합친 후에 팀별로 전처리, 모델링과 Backend, product serving 등을 순차적으로 해 나갔다. 하지만 의사결정이 필요할 때는 다 같이 이야기를 나누었고, 방향성은 함께 잡아나갔다. 마지막에는 전체적으로 서버에 올린 후에 제대로 작동하도록 에러를 고치고 UI 개선과 두 모델을 합쳐서 사용하는 등 완성도 있는 서비스를 위해서 노력했다. Product serving 시에 필요한 전체적으로 서비스도 성공적으로 만들고 Backend와 product serving을 경험하고 적용하고자 하는 목표를 달성했다.

2. 나는 어떤 방식으로 서비스를 개선(향상)시켰는가?

Frontend를 구현하기 위해서 Streamlit를 사용해서 한 페이지에 모든 요소들을 나열한 후 백엔드와 기능적으로만 연결한 프로토타입을 만들었다. 사용자의 입장에서 해당 화면이 바뀌는 것이 단계를 지나고 있다는 인식을 줄 수 있다고 생각하여 session state를 사용해서 페이지를 바꾸는 것과 같은 효과를 주었다. 또한, 설문조사 응답에 서비스 이용 시에 개선되었으면 하는 사항들을 받아서 streamlit 내에서 구현할 수 있는 것들을 최대한 반영했다. 홈버튼, 사진 노출 방식, 안내문 등을 추가하여 서비스를 이용할 때 헛갈리지 않게 제대로 사용하고, 보여지는 것도 잘 보일 수 있게 노력했다.

GCP에서 서버를 할당받아 백그라운드에서 서버를 활성화시켜 실시간으로 제공할 수 있도록 했다. 가장 먼저 DB 서버를 할당받아 DBBeaver와 연결한 후 데이터 스키마를 설계하여 데이터를 적재했다. 이후 FastAPI와 pymysql을 사용하여 추천된 아이템의 정보를 서버에서 받아올 수 있게 했다. 또한, GCP에서 VM서버를 할당받아 backend server와 frontend server를 구축했다. Backend 서버에서 FastAPI를 사용해서 DB 서버와 데이터를 주고받으면 Frontend server의 서비스에 결과를 보여주는 역할을 하도록 했다.

DB를 GCP의 서버에 올리기 위해서 논리적, 개념적 모델링과 같은 데이터 스키마를 정해보며 데이터 설계를 직접 경험했고, API 명세서를 적어서 구현하기 전에 필요한 고려 사항들을 정리한 후에 DB를 만들었다, DB를 서버에 올린 이후 다른 기능이 추가될 때마다 계속해서 API가 필요했고, 기능을 FastAPI로 추가적으로 구현했다.

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

프로젝트를 여러 번 진행하면서 내가 구현하려는 코드를 어디에 써야 하는 지도 한 번 더 고려하면 협업하기 좀 더 원활해 진다는 것을 깨달았다. 그래서 이미 만들어진 파일에 내가 코드를 덧붙일 때 그 파일을 만든 팀원에게 한 번 더 확인한 후에 구현했다. 또한, 나 스스로 제대로 된 위치를 찾기 위해서는 파일별로 용도를 이해해야 했기에 각 파일이 하는 기능의 역할에 대해서 생각한 후 개념도로 그려보았다. 그랬더니 조금 더 구조화되어 알맞은 곳에 대한 생각이 명확해졌고, 정돈된 코드를 만들 수 있었다.

GCP를 처음 사용해 보면서 관련 코드를 documentation에서 찾아서 구현했다. “일단 실행되게 한 후에”라는 급한 마음을 가지고 하니, 문제들이 꼬였다. 오히려 시간을 가지고 내가 지금 실행하고 있는 코드가 어떤 기능을 해서 이 documentation에서 사용하라는 것인지 한 번 꼭 꼼꼼하게 읽고 실행하는 것이 시간도 아끼고 이해하며 할 수 있었을 것이라고 생각했다. 그래서 처음부터 차근차근 documentation을 놓치지 않고 읽으며 했더니 에러도 사라지고 문제를 해결할 수 있었다.

4. 마주한 한계/교훈은 무엇이며, 다음 프로젝트에서 스스로 새롭게 시도해볼 것은 무엇일까?

Backend 서버를 할당받은 사용자만 그 파일들에 접근해서 수정하고 실행하는 access 문제에 맞닥뜨렸는데, 구글링을 통해서 해결하지 못하고 프로젝트를 마무리해서 많이 아쉬웠다. Product serving을 하면서 사소한 것 하나라도 고칠 때마다 다시 코드를 하나씩 업데이트해줘야 하는 상황을 경험하며 자동화의 필요성을 몸소 느낄 수 있었다. 그래서 CI/CD와 Airflow, Kubernetes 등을 추가로 도입해 보면 좋을 것 같다.

1. 최종 프로젝트의 목표

<What's In Your Closet?> 이라는 제목으로 진행된 최종프로젝트는 사용자가 가지고 있는 옷을 입력하면 그와 잘 어울리는 다른 옷이나 코디를 추천해주는 서비스이다. 나는 이 서비스를 처음부터 만들어가면서 AI서비스가 어떻게 만들어지고 실제 사용자에게 어떤 식으로 배포가 되는지 그 전반적인 과정을 경험하고 싶었다. 따라서 최종 프로젝트의 완성도도 중요한 목표로 삼았지만, 이런 머신러닝 라이프사이클을 한 번 경험해보고자 하는 것에 중점을 두었다.

2. 프로젝트를 개선시킨 방법 (모델, 데이터, 서빙)

나의 역할은 크게 데이터의 수집과 전처리 그리고 추천을 진행하는 핵심 파트인 모델의 설계와 UI(프론트엔드)의 개선이었다.

2-1. 데이터 크롤링

이전에 웹 개발을 여러 번 진행하면서 HTML에 대한 이해도가 있던 덕분에 처음 사용해보는 Selenium을 통한 크롤링도 원활하게 잘 할 수 있었다. 다만 어떤 데이터를 더 효과적으로 크롤링하고 어떤 정보들을 가져와야 하는지 고민을 많이 했었다. 특히 처음부터 Light GCN이라는 모델을 사용하기로 했기에 수집하는 데이터 간의 연관성이 매우 중요하게 작용되었다. 이 연관성을 데이터 속에서 찾기 위해 크롤링 단계에서부터, 코디 내에 있는 모든 아이템들을 수집하고 여기서 코디와 아이템의 관계, 그리고 그 아이템 간의 관계를 이용하기로 하였다. 그리고 다시 아이템들은 다른 코디와 연결되어 있는데, 이 코디들을 다시 크롤링 하는 어떻게 보면 DFS와 같은 크롤링 방식을 선택하였다. 덕분에 모델링을 할 때, 꽤 좋은 결과를 얻어낼 수 있었다.

2-2. 모델링 (Light GCN)

Light-GCN을 구현할 때는 저번에 모듈화해서 정리해둔 코드를 가져와서 조금 수정만 하면 되었기에, 매우 간편하게 실험을 할 수 있었다. 이 LightGCN은 노드와 노드 사이를 연결하고 그 간선을 관련됨(1), 관련되지 않음(0)으로 라벨링을 진행해야 하는데, 관련되지 않음(0)을 어떻게 정해야 할지 깊은 고민을 하였다. '관련됨'은 크롤링할 때부터 코디-아이템 간의 관계를 파악했기 때문에 그대로 사용하면 되지만, 관련되지 않음은 서로 연결이 되어있지 않은 코디-아이템을 사용하기에는 위험하다는 판단이 들었다. 실제로 해당 아이템이 코디에 등장했을 수 있는데 크롤링 결과에 없다는 이유만으로 관련되지 않았다고 하기에는 위험하다는 생각이 들었다. 이번 프로젝트에서 직접적으로 이 문제를 해결하지는 못했지만, 간접적으로 이 Negative Sampling의 비율을 조절하여 균형 있게 학습을 하도록 유도했다.

2-3. Front-End (Streamlit)

팀원이 완성시킨 Front-End UI에 UX를 고려하여 어떤 식으로 버튼을 배치할지, 어떤 정보를 보여줘야 사용자가 만족스럽고 신뢰할 수 있을지 고민을 해보았다. 실제로 처음에 1차로 배포된 프로토타입에서 UI/UX와 관련된 피드백이 많았기에 반드시 해결해야하는 문제라고 정하였다. 대표적으로 기존에 검색하기 어렵다는 피드백을 받아, 검색어를 수정하기도 했으며 사용자에게 더 신뢰감 있는 추천 결과를 제공하기 위해 모델이 몇 %의 확률로 어울릴 것이라고 판단했는지 보여주어 수치적인 정보를 통해 신뢰감을 주고자 했었다.

3. 내가 한 행동의 결과와 깨달음

어떤 프로젝트를 진행하더라도 협업을 하게 되었을 때, 협업 능력을 가장 많이 끌어올릴 수 있는 것은 실력보다는 소통이라고 생각된다. 그래서 이 소통을 통해서 꾸준히 다른 팀원들의 진행사항이 어떻게 되는지 어떤 문제가 있는지 공유하고자 했었다. 하나의 메타버스인 '게더타운' 서비스를 이용하며 언제든지 소통이 가능하도록 했고 결과적으로 빠른 피드백을 통해서 시간낭비 없이 더 많은 시도들을 할 수 있게 되었다.

이번 프로젝트에서도 깔끔한 코드를 작성하기 위해서 노력을 들었다. 누구나 접근해서 수정할 수 있거나 모듈화해서 사용할 수 있도록 만들었는데, 특히 데이터 크롤링을 진행할 때 정말 유용하였다. 끊임없는 고민을 하면서 크롤링 방법이 정말 여럿 수정되었는데 그때마다 모듈화 한 함수만 수정하면 되어서 큰 편리함을 안겨다 주었다. 처음에는 모듈화 하는 것이 어색하고 복잡하였는데 하다 보니 스스로도 그 이점을 많이 터득하고 클린 코드를 작성하는 능력을 기르게 되었다고 생각하였다.

4. 한계와 아쉬운 부분

사용자에게 좋은 서비스를 제공하기 위해서 좋은 성능(결과)을 보여주는 모델이 중요한데 그 전에 더 큰 데이터셋이 있으면 성능을 분명 높일 수 있었다고 생각한다. 특히 이번 모델링 관점에서 가장 신경을 많이 쓴 부분이 상호작용 행렬의 sparsity 문제였다. 이 sparsity가 매우 높다 보니 여러 모델을 사용하기에 한계가 있었고, 모델을 사용하게 되더라도 만족스러운 결과를 얻지 못했었다. 이는 더 많은 양의 데이터를 크롤링하면 충분히 개선할 수 있는 문제라고 생각된다.

5. 개선사항, 새롭게 시도해볼 수 있는 것들

크게 느꼈던 것은 사용자는 의외로 성능에 대해 평가하기전 UI/UX에 대해 관심이 많다는 것을 알게 되었다. 그렇기에 더 편리한 UI/UX를 주고자 프론트를 개선시키고 싶으며, 데이터 수집과 전처리 그리고 DB저장 과정을 하나의 프로세스로 만들어주는 AirFlow를 활용하면 개발 cost를 많이 줄일 수 있을 것 같다. 추가로 도커나 쿠버네티스를 사용해 배포 최적화도 해보고 싶다.

1. 최종 프로젝트의 목표

개인 학습 목표는 이커머스 플랫폼 서비스를 구현하며 실제로 현업에서 발생할 수 있는 문제를 최대한 경험해보고, 데이터 엔지니어링부터 MLOps까지 여러 분야에 참여해보는 것으로 정했다. 팀의 공동 학습 목표는 프로젝트의 전체적인 Life Cycle을 경험하고 Serving 되었을 때, 사용자가 사용하기 쉽고 만족할 UI를 구현하는 것으로 결정했다. 개인 학습 측면에서는 데이터 수집과 전처리를 담당하며 자동화 파이프라인을 구축했고, GCP 서버에 데이터베이스를 설계하고 관리하며 데이터 흐름을 파악했다. 또한, GCP Back-end 서버에서 FastAPI를 사용해 기능을 추가해보며 Product Serving 과정을 이해하고자 했다. 공동 학습 측면에서는 Jira와 Confluence 협업 툴을 도입하여 적극 활용하였다. Jira에서 각 팀원별 에픽과 하위 이슈를 등록하여 일정 관리를 했고, 프로젝트 진행 상황을 Confluence wiki를 통해 체계적으로 관리했다.

2. 서비스 개선 방법

2-1. 데이터 크롤링

패션사이트인 '무신사'에서 크롤링을 통해 패션 아이템과 코디 데이터를 각각 수집했다. 수집하기 앞서 데이터 탐색을 통해 수집해야 할 중요한 정보(feature)를 결정했고, 추후에 DB에 데이터를 적재할 때 어떻게 저장하고 관리해야 할지 고려하며 데이터 구조를 설계했다. Python의 Selenium과 Chrome Driver를 통해 크롤링을 진행하였고, 옷과 코디 사이의 상호작용 정보를 최대한 늘리기 위해 아이템(옷, 코디)을 수집할 때 연관된 아이템(옷, 코디)을 우선으로 크롤링하는 전략을 사용했다. 예로, 하나의 코디 내에 여러 옷들이 존재하면 각각의 옷들과 연관된 다른 코디들을 다시 크롤링했다.

2-2. 데이터 전처리

아이템 각 속성의 결측치와 변수 타입 및 종류(연속형, 카테고리형)를 분석하고 시각화하여, 팀원들이 결측치나 이상치 처리에 대해서 의사결정을 할 때 참고할 수 있도록 했다. EDA를 통해 파악한 각 데이터의 속성을 고려하여 전처리하는 코드를 구현했다. 특히 아이템 속성간의 연관성을 파악하고 결측치를 다른 속성을 통해 채울 수 있도록 했다. 또한, 기존 정보에서 새로운 정보를 생성하는 Feature Engineering을 진행하였다. 마지막으로, 전처리 순서를 정의하고 데이터 파이프라인을 설계하였다

2-3. Back-end (GCP, MySQL DB, FastAPI)

GCP 서버에 MySQL DB를 생성하고 dbeaver를 통해 서버와 연결하여 지속적으로 업데이트 했다. 데이터 간의 연관성을 고려하여 데이터베이스 스키마를 설계하고, 데이터를 적재했다. 백엔드에서는 사용자가 선택한 아이템의 중분류와 추천된 아이템의 cluster Id를 DB에서 가져올 수 있도록 API를 구축하고 프론트엔드에서 사용할 수 있도록 했다. 또한, 아이템의 사이트 주소를 DB에서 가져오는 API를 만들어 웹 사이트에서 아이템의 이미지를 클릭하면 해당 아이템의 상품 페이지로 이동할 수 있도록 구현했다.

2-4. UI 개선

Streamlit 프론트엔드 서버에서 서비스를 제공했다. 아이템이 추천되었을 때 아이템이 LightGCN 모델의 추천 확률로 정렬되어 전시될 수 있도록 구현했다. 또한, 아이템 추천 결과가 사용자가 선택한 아이템과 같은 대분류에서 추천되지 않도록 대분류 중복 추천을 제거하는 기능을 추가했다. 이외에도 1차 설문 피드백을 UI에 반영하여 사용자의 편의성을 개선시켰다.

3. 수행 성과 및 깨달음

전략적인 데이터 수집과 전처리를 통해 잘 정제된 데이터를 얻었고, 이를 통해 모델의 성능을 높일 수 있었다. 초반에 데이터 저장 경로와 구조를 정해놓아 수월하게 데이터 수집을 진행할 수 있었다. 또한, 1차 사용자 피드백을 서비스에 많이 반영하였는데, 사용자 관점에서 또 놓친 부분은 없는지 생각하게 된 계기가 되었다. 추천된 결과 정보가 미흡하게 제공되고 있다는 생각이 들었고, 해당 아이템의 이미지를 눌렀을 때 상품 페이지로 이동하게 구현함으로써 더 나은 서비스를 제공할 수 있었다. 결국 서비스는 사용자가 이용하는 것이므로 항상 고객 관점으로 생각할 수 있는 개발자가 될 수 있도록 노력해야겠다고 생각했다.

4. 새롭게 시도한 변화

베타 서비스를 배포하여 사용자에게 서비스를 제공하고 중간 설문 피드백을 받았다. 개발자 입장에서 생각해보지 못한 다양한 피드백을 받았고, 이를 최대한 반영하며 시야를 넓힐 수 있었다. 또한, 이번 프로젝트에서 기능을 구현할 때 기능 별로 모듈화해서 타입 힌트나 주석을 통해 다른 팀원도 코드를 쉽게 이해할 수 있도록 노력했다.

5. 한계 및 아쉬웠던 점

이번 서비스의 주 기능 중 하나인 코디 추천 페이지에서는 사용자가 선택한 아이템과 추천된 아이템으로 조합된 코디 정보를 제공하는데, 두 아이템 간의 상호작용이 없을 때는 코디 정보를 제공하지 못한다는 한계점이 존재했다. 이러한 문제 때문에 다양한 AI 모델을 시도해보지 못한 점과 룰베이스 모델과 LightGCN만을 사용했기 때문에 아이템의 다양한 속성을 활용해보지 못한 점이 아쉬웠다. 또한, 주어진 시간이 너무 짧아서 목표로 정했던 Airflow나 Kubernetes를 사용해보지 못한 것도 아쉬움이 남는다.

1. 최종 프로젝트의 목표

이번 프로젝트의 목표는 '사용자가 가지고 있는 옷을 최대한 활용할 수 있는 옷을 추천하고 그에 맞는 코드를 추천하는 서비스를 만드는 것' 이었다. 이 과정을 통해 AI 모델을 서비스에 적용하기 위해 필요한 기술, 그 과정에서 발생하는 문제를 해결하는 역량을 갖추는 것이었고 이번 프로젝트를 통해 실제 현업에서 문제를 해결하기 위해 사용하는 기술을 응용해서 프로젝트를 완성하고자 하였다.

2. 프로젝트를 개선시킨 방법 (서버 구축, 프론트 페이지 구현, 배포)

이번 프로젝트에서 내 역할은 프론트엔드 서비스를 구축하고 이때 필요한 주 기능을 백엔드 서버를 구축해서 백엔드 서버와 프론트엔드 페이지가 API를 통해 데이터를 주고받고 동작할 수 있도록 만드는 것이었다.

2-1. FastAPI 백엔드 서버 구축

동영상 강의를 통해 얻은 지식과 기존에 가지고 있던 백엔드 기술을 활용해 FastAPI 서버를 구축할 수 있었다. FastAPI를 사용하면서 느낀 것은 간단하게 활용할 수 있지만 매우 강력한 기능을 제공한다는 점이였다. FastAPI를 통해 빠르게 API 기능을 구축할 수 있었다. 이번 프로젝트에서는 모듈화된 서비스를 구현하고 싶었다. 따라서 API controller, DB connector, CRUD 기능을 최대한 쪼개고 모듈화해서 프로젝트를 확장가능하게 만들 수 있도록 설계하였다.

2-2. Streamlit 서버 구축

Streamlit을 통해 프론트엔드 페이지를 구축하였다. Streamlit은 디자인에 있어 극도로 제한된 기능만을 제공했기 때문에 어떻게 해야 사람들에게 좋은 UI/UX를 제공할 수 있을지 많은 고민을 하였고 사용자들에게 피드백을 받아 완성도 있는 화면 디자인을 만들 수 있었다. Streamlit기능에서 페이지를 이동하는 기능을 구현하기 위해 Session State라는 전역 변수를 사용해서 전역 변수 조건에 맞게 화면을 변경하도록 구현하였다. Grid 형태로 옷 이미지를 배치하기 위해 Column 모듈을 사용해 10개의 이미지를 2 X 5 형태로 배치할 수 있었다.

2-3. GCP 서버 세팅 및 서버 배포

GCP에서 사용한 주 서비스는 Computing Engine과 SQL Database 서비스이다. Computing Engine에는 conda 가상환경을 설정하고 서버를 구동하였다. 이후 방화벽 포트를 오픈하여 외부에서 서버로 접속할 수 있도록 환경을 구축하였다. Streamlit과 FastAPI는 서비스를 간단하게 실행할 수 있기 때문에 빠르게 서비스할 수 있는 환경을 만들 수 있었다. SQL은 MySql 서버를 클릭 몇 번으로 만들 수 있었다. DBbeaver라는 DB 접속 툴을 통해 DB에 접속하여 스키마를 생성하고 테이블을 생성하였다.

3. 내가 한 행동의 결과와 깨달음

- 서비스 할 수 있는 형태로 프로젝트를 완성

이번 프로젝트를 웹으로 사용자들에게 배포할 수 있게 되었다. 기본적인 서비스 구조 프론트-백엔드-DB 를 구축하여 추천 모델을 서빙할 수 있게 만들 수 있었다.

- 개발자로서 고민해야하는 부분

AI 서비스를 만들기 위해 개발자로서 고민해야하는 것들이 너무 많고 배워야하는 기술이 산더미 같다는 것을 깨달았다. 단순한 기능도 실제로 구현하기 위해서는 많은 기본적인 지식과 문제해결능력을 갖춰야함을 알 수 있었다.

4. 한계와 아쉬운 부분

프로젝트 시작을 하는 시점에 팀원들과 새로운 목표에서 많이 벗어나게 된 점이 아쉬웠다. 특히 이번 프로젝트에서 꼭 해보고 싶었던 Airflow, 쿠버네틱스를 사용해 보지 못한 부분이 아쉬웠으며, 모델링에 있어서도 LightGCN과 Rule base모델로 추천 결과를 서빙했던 부분도 아쉽다.

5. 개선사항, 새롭게 시도해볼 수 있는 것들

사용자들의 피드백을 통해 얻은 문제점들을 개선하고, 이번 프로젝트 기간동안 구축하지 못한 데이터 파이프라인, 모델 파이프라인을 구축해볼 수 있을 것 같다. 사용자의 Log data를 수집해 추천 결과에 반영할 수 있는 피드백 루프를 만들어 서비스를 개선 시킬 수 있을 것이라 생각한다.

1. 최종 프로젝트의 목표

이번 프로젝트의 목표는 '사용자가 가지고 있는 옷을 최대한 활용할 수 있는 옷을 추천하고 그에 맞는 코드를 추천하는 서비스를 만드는 것' 이었다. 이 과정을 통해 AI 모델을 서비스에 적용하기 위해 필요한 기술, 그 과정에서 발생하는 문제를 해결하는 역량을 갖추는 것이었고 이번 프로젝트를 통해 실제 현업에서 문제를 해결하기 위해 사용하는 기술을 응용해서 프로젝트를 완성하고자 하였다.

2. 프로젝트를 개선시킨 방법 (데이터 수집 및 전처리, 모델링)

2-1. Selenium Data Crawling

Selenium이라는 동적 크롤러를 사용하여, 무신사에 있는 코드 정보와 아이템 정보를 수집했다. 처음에는 페이지 단위로 최신 코드를 수집했다. 하지만 최신 코드는 독특한 아이템이 너무 많이 포함되어 있었으며, 페이지 단위로 크롤링하게 되면 아이템 연결 관계가 너무 Sparse 해진다는 문제가 있었다. 이러한 문제들을 해결하기 위해 우선 코드를 인기 순으로 정렬한 후 크롤링 했고, 페이지 단위가 아니라 관계 깊이 단위로 크롤링을 했다. 관계 깊이 단위로 크롤링했다는 것은, 우선 1차적으로 코드와 포함되어 있는 아이템을 수집하고, 크롤링한 아이템이 포함되어 있는 타 코드를 확인하여 추가 크롤링을 반복하는 것을 말한다. 이러한 방법들로 데이터 sparsity 문제를 해결할 수 있었다.

2-2. Data Preprocessing

데이터를 수집한 후, 어떻게 데이터를 전처리 할지 고민했다. 예를 들어 '좋아요 수'와 같은 데이터는 Int 형식으로 바뀌었으며, 결측치는 좋아요가 아직 없는 것으로 판단하여 0으로 채워 넣었다. 또한 중분류와 대분류는 일정한 형식에 맞도록 변경해주었다. 데이터를 전처리 했을 뿐만 아니라 수집한 데이터를 사용하여 새로운 feature를 만들기도 했다. 동일한 중분류와 색을 갖고 있는 아이템들을 하나의 cluster로 묶고, cluster ID를 부여한 것이 예이다. 이를 사용하여 아이템 상호작용 행렬이 아닌, 클러스터 상호작용 행렬을 만들어 데이터 sparsity를 더 낮출 수 있었다.

2-3. Rule-Based Model

데이터 sparsity가 너무 높았기 때문에, 머신러닝, 딥러닝 모델들이 학습을 제대로 하지 못할 것이라고 생각했다. 따라서 안정적인 추천을 하기 위해 Rule-based model을 구현하기로 결정했다. Rule based model은 item ID를 받아서, 해당 아이템이 포함되어 있는 클러스터를 찾고, 상호작용한 타 클러스터 안에 있는 모든 아이템을 추천 후보 아이템 집합에 넣는다. 마지막으로, Light-GCN 모델에서 나온 상호작용 확률이 높은 10개의 아이템을 추천한다.

3. 내가 한 행동의 결과와 깨달음

- 모델과 데이터의 땀 수 없는 관계

이번 프로젝트에서 데이터 엔지니어링과 모델링을 담당하면서, 이 둘은 서로 땀 수 없는 관계임을 실감했다. 데이터를 수집하고 전처리 하기 위해서는 모델링을 이해하고 있어야 하고, 반대로 마찬가지로였다. 이 두 분야가 더 긴밀하게 협업할수록 일의 효율과 모델의 성능이 개선되는 것을 느꼈다.

- 유연한 개발의 중요성

상황에 맞게 적절히 서비스를 계획하고 구현해야 된다는 것을 깨달았다. 최종 프로젝트는 지금까지 배운 것을 종합적으로 응용할 수 있는 기회였기 때문에, 사실 더 복잡한 최신 모델을 사용하고 싶었다. 하지만 우리가 크롤링한 데이터는 복잡한 모델이 학습하기에 충분한 정보를 담고 있지 않았기 때문에, 오히려 성능이 떨어지는 역효과가 날 수 있었다. 따라서 우리 팀은 규칙 기반 모델이 더 좋은 성능을 보장한다고 판단하여 이를 메인 모델로 채택했다. 결과적으로 규칙 기반 모델은 안정적이면서 좋은 성능을 냈다.

4. 한계와 아쉬운 부분

Airflow, Kubernetes와 같이 더 다양한 백엔드 툴들을 사용하지 못한 것이 아쉬웠다. 또한 프론트 앤드를 더 고도화 시키지 못한 것이 아쉬웠다. 시간이 더 주어진다면 전반적으로 서비스를 개선할 수 있는 여지가 많다고 느꼈다.

5. 개선사항, 새롭게 시도해볼 수 있는 것들

사용자 설문에서 공통적으로 나왔던 개선사항은 아이템의 종류가 부족하다는 점과 웹 페이지가 사용하기 불편한 점들이 있다는 점이었다. 앞으로 Airflow를 통해 데이터 수집 및 처리 작업을 자동화하여 더 다양한 데이터를 수집하여 서비스의 질을 개선하고, 웹 페이지 기능들을 추가하여 더 사용하기 편한 서비스를 구현하고 싶다.