2024 가톨릭대학교 인근 지역 음식점 추천 서비스

컴퓨터정보공학부 고경우,이윤서

Catholic Univerisity of Korea

CONTENTS

01

서비스 소개.

해당 대회에서 제시하는 문제에 대한 인식과 중요 시사점 파악,
만들어낸 서비스 소개

01

문제 제시

본교 인근 지역 음식점 정보와 직/간접적 사용자 피드백 정보를 수집하여,
사용자에게 개인화된 음식점 추천을 제공할 수 있는 모델 및 시스템 개발

사용자 피드백 : 점진적 고도화

개인화 : 보편적인 아닌 개인에 맞춤화

시스템 : 데이터의 처리만이 아닌 수집할 생태계

02
냠톨릭2.0

리뷰기능과 사용자 맞춤 추천 기능이 적용된 가톨릭대학교 인근 음식점 안내 사이트

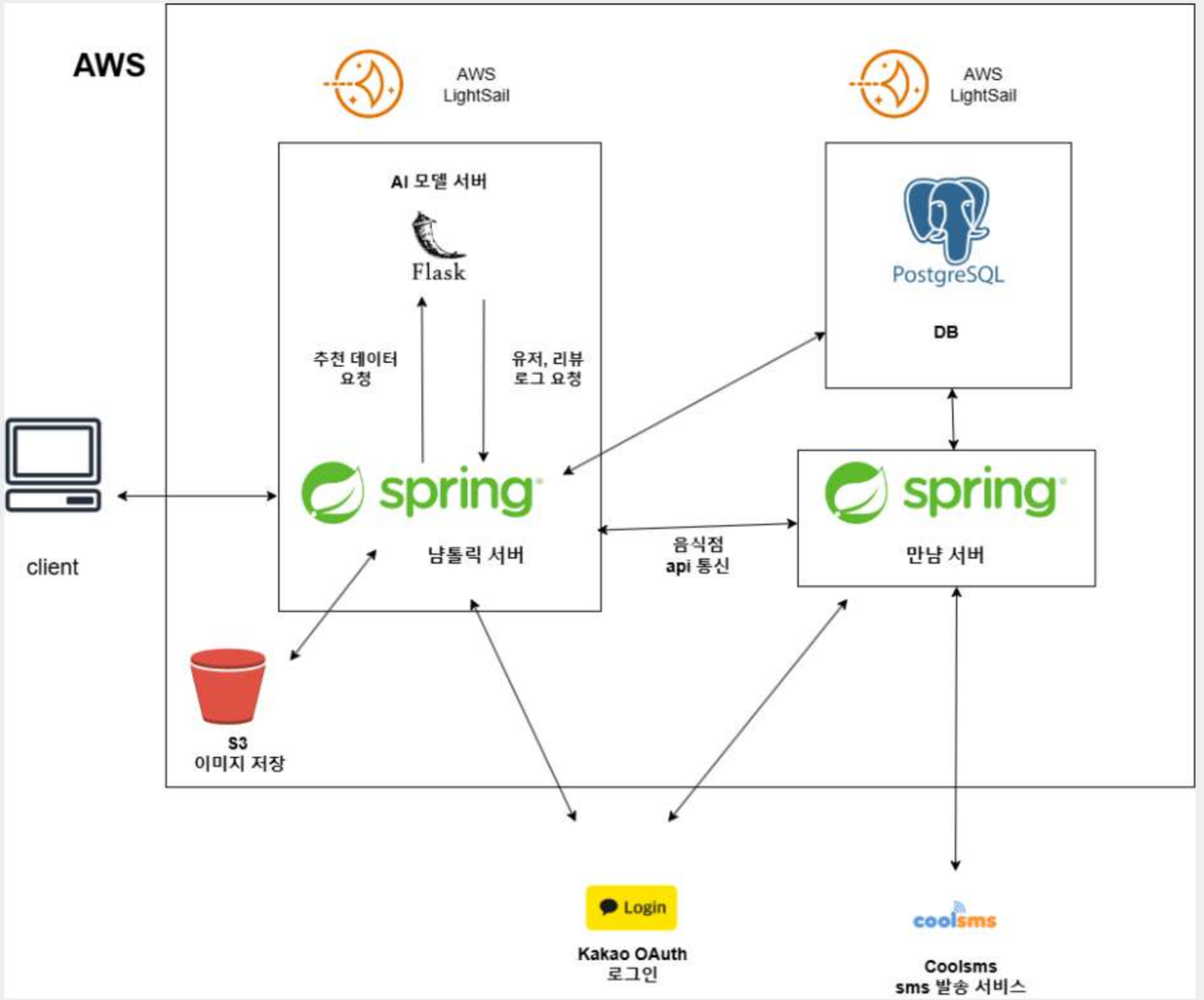
냠톨릭은 가톨릭대 인근의 음식점에 대한 정보를 제공해주는 웹사이트입니다.
음식점들의 평점과 위치, 전화번호, 메뉴, 특정 카테고리를 알려주는 서비스로 매일 50명 이상이 방문하는 독자적인 서비스입니다.
이번 대회를 통해 머신러닝 기반 사용자 맞춤 추천 서비스를 도입하면서 냠톨릭2.0의 버전을 새로이 보여드립니다.

02

시스템 아키텍처.

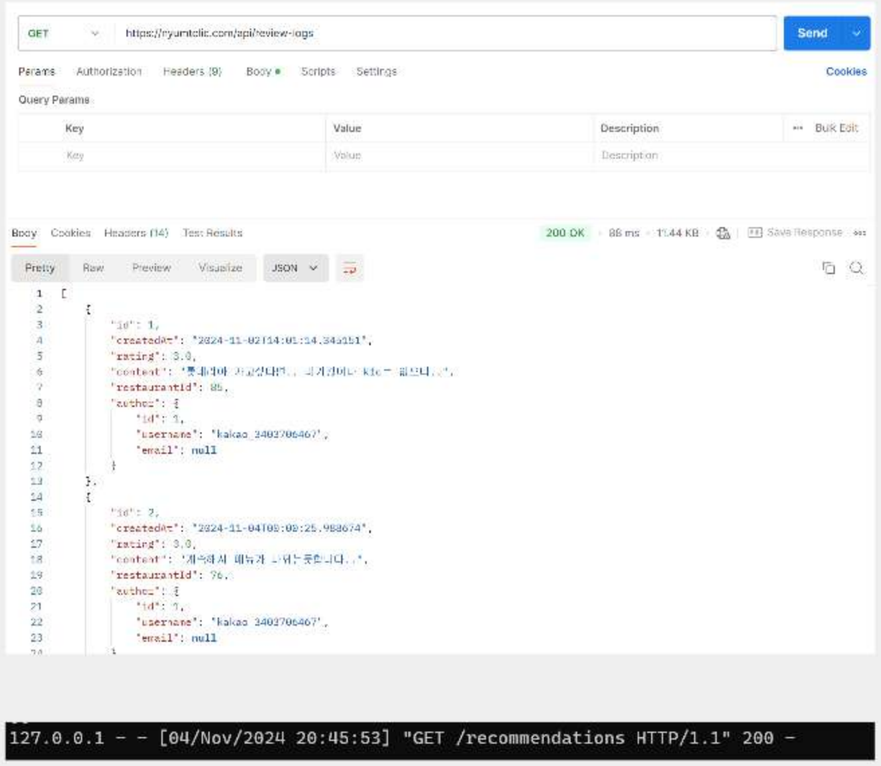
배포와 런칭이 완료되어 실제 웹상에서 사용되는 서비스,
서비스의 설계와 데이터베이스 구성

아키텍쳐 다이어그램



냠톨릭 Spring Boot 서버 / AI 추천 모델 서버 (Flask 기반) / PostgreSQL데이터 베이스 / lightsail

아키텍쳐 다이어그램



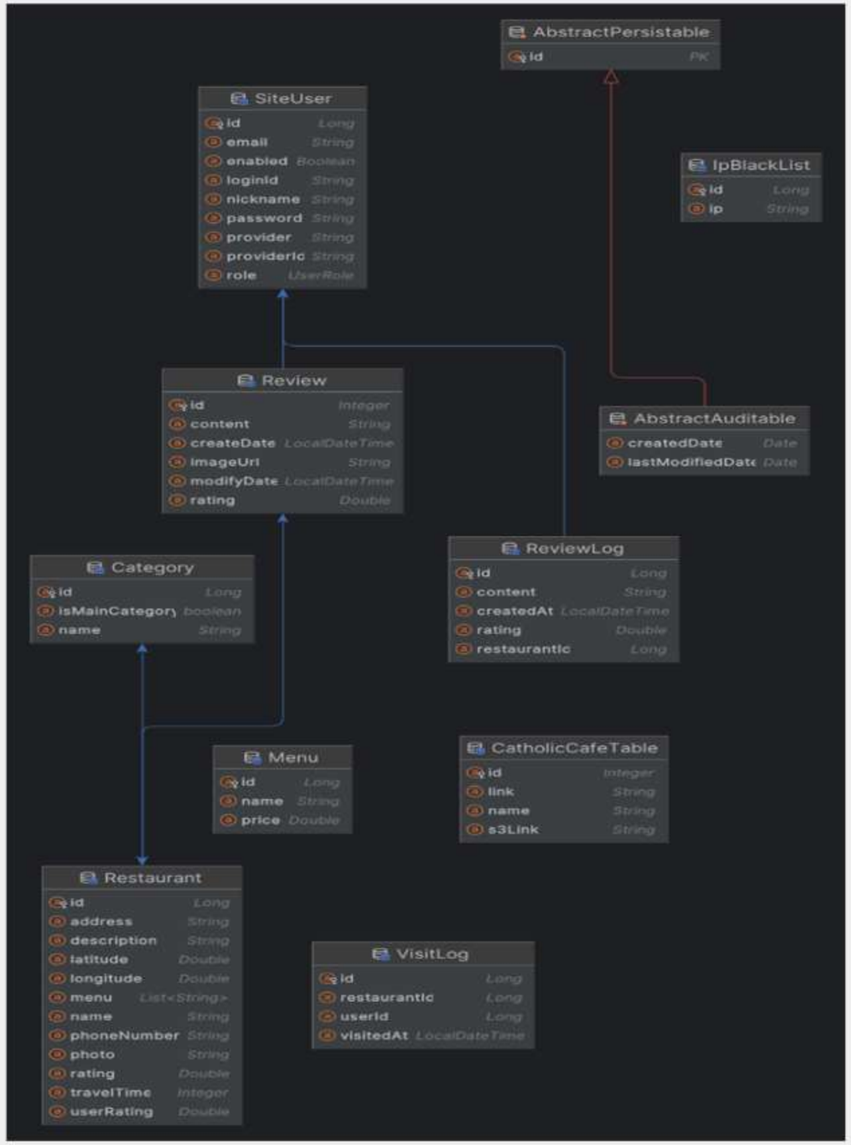

```
127.0.0.1 - - [04/Nov/2024 20:45:53] "GET /recommendations HTTP/1.1" 200 -
```

Spring Boot로 구성된 메인 애플리케이션 서버와 Flask로 구성된 AI 모델 서버 간의 REST API 통신을 통해 사용자 맞춤형 추천 서비스를 제공합니다.

이 두 서버는 HTTP 요청을 주고받으며, 각자의 역할에 맞는 데이터를 상호 교환하여 사용자에게 음식점 추천을 제공합니다.

postman을 사용하여 개발진이 구현한 db와 서버간의 통신 로직 api를 확인했습니다.
아래 get 요청시에 200을 보이며 원활히 통신되는 것을 확인할 수 있습니다.

ERD


AbstractPersistable
id PK
SiteUser
id Long
email String
enabled Boolean
loginId String
nickname String
password String
provider String
providerId String
role UserRole
IpBlackList
id Long
ip String
Review
id Integer
content String
createDate LocalDateTime
imageUrl String
modifyDate LocalDateTime
rating Double
AbstractAuditable
createdDate Date
lastModifiedDate Date
Category
id Long
isMainCategory boolean
name String
ReviewLog
id Long
content String
createdAt LocalDateTime
rating Double
restaurantId Long
Menu
id Long
name String
price Double
CatholicCafeTable
id Integer
link String
name String
s3Link String
Restaurant
id Long
address String
description String
latitude Double
longitude Double
menu List<String>
name String
phoneNumber String
photo String
rating Double
travelTime Integer
userRating Double
VisitLog
id Long
restaurantId Long
userId Long
visitedAt LocalDateTime

03

서비스 계층 구현.

냠톨릭이 제공하는 주요 기능의 페이지별 설명

01 서비스 주요 페이지 설명

음식점 리스트 페이지

남톨릭2.0 음식점 랜덤 추천 음식점 리스트 학식정보 만남 건의사항 로그인 회원가입

남톨릭-가톨릭대 주변 음식점 사이트

음식점 리스트

전체 중식 양식 한식 분식 일식 카페 술 디저트 고기 닭 배달 데이트 혼밥

이름순 유저 평점 순

1983 PIZZA&PUB
사각피자
양식
3.0
4.5

60계 치킨
크크크치킨
고기 닭 배달
3.0
0.0

88닭발
88닭발
술 닭
3.0
0.0

THE 53
제육
한식 혼밥
4.0
4.8

bbq
황금 올리브 치킨
고기 닭 배달
3.0
0.0

음식점 상세 페이지

냠들릭2.0 음식점 랜덤 추천 음식점 리스트 학식정보 만냠 건의사항 로그인 회원가입

까르디 에스프레소 바

★★★☆☆ 3.6

★★★★☆ 4.6

주소: 경기 부천시 원미구 부일로 705

전화번호: 0507-1337-9524

소요 시간: 6분

설명: 분위기가 모던한 에스프레소 바. 메뉴 하나하나가 다 맛있어요.

카테고리: 디저트, 카페, 데이트

대표 메뉴: 에스프레소, 샤케라또

고경우 아님

☆☆☆☆☆

샤케라또 매우맛있습니다

작성 시간: 2024-11-02 13:55

추천 0

떵개

☆☆☆☆☆

진짜 자주가요 샤케라또 추천

작성 시간: 2024-06-10 22:27

추천 0

음식점 랜덤 추천 페이지

남돌릭2.0 음식점 랜덤 추천 음식점 리스트 학식정보 만냠 건의사항 로그인 회원가입

남돌릭-가톨릭대 주변 음식점 사이트

신전떡볶이

★★★☆☆ 3.0

☆☆☆☆☆ 0.0

주소: 경기 부천시 원미구 부일로 695

전화번호: 032-341-1120

소요 시간: 5분

설명: 여러분이 잘 아시는 신전 떡볶이. 자극적인 떡볶이를 원하시면 추천해요. 식사 생분들은 1인 메뉴도 추천드릴게요.

카테고리: 분식, 혼밥

대표 메뉴: 떡볶이, 튀김

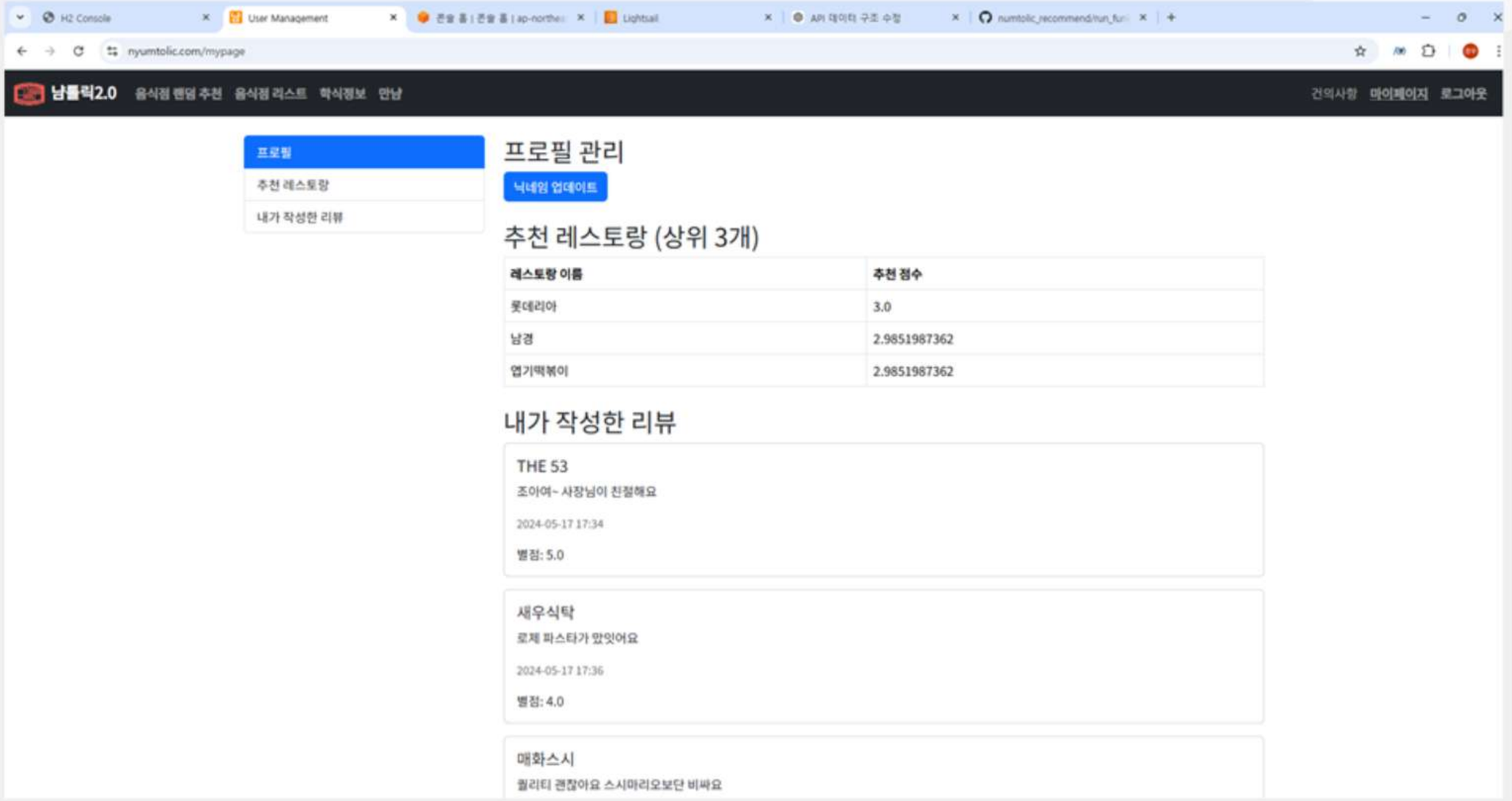
제외할 카테고리를 골라주세요:

제외할 카테고리를 아래에서 골라주세요!

중식 양식 한식 분식 일식 카페 술 디저트 고기 닭 배달 데이트 혼밥

음식점 랜덤 추천

마이페이지


H2 Console
User Management
nyumtolic.com/mypage
남톨릭2.0
음식점 랜덤 추천
음식점 리스트
학식정보
만남
건의사항
마이페이지
로그아웃
프로필
추천 레스토랑
내가 작성한 리뷰
프로필 관리
닉네임 업데이트
추천 레스토랑 (상위 3개)
레스토랑 이름
추천 점수
롯데리아
3.0
남경
2.9851987362
엽기떡볶이
2.9851987362
내가 작성한 리뷰
THE 53
조아여~ 사장님이 친절해요
2024-05-17 17:34
별점: 5.0
새우식탁
로제 파스타가 맛잇어요
2024-05-17 17:36
별점: 4.0
매화스시
퀄리티 괜찮아요 스시마리오보단 비싸요

학식 정보 페이지

구분		11/04(월)	11/05(화)	11/06(수)	11/07(목)	11/08(금)
점심	Global Noodle	온메밀국수	해물볶음우동	전주식돌계탕국수	(냄비)얼갈라면	꼬치어묵잔치국수
		김가루밥	추가밥/명이미소국	추가밥	추가밥	후리가케밥
		황당두팸	생선까스*타르소스	너비아니*야채무침	타코야끼	오징어링*감자튀김/요구르트
		단무지산고추	피클단무지	반달단무지	짜사이단무지	콩나물무침
		배추김치	배추김치	배추김치	배추김치	깍두기
		814 kcal	737 kcal	879 kcal	942 kcal	887 kcal
	Yummy 덮밥	아비꼬st카레덮밥	닭갈비덮밥	베이컨락두기볶음밥*계란후라이	계란볶음밥*마파두부소스	중화풍제육덮밥
		우동국물	명이미소국	어묵국	유부장국	미역국
		통등심돈까스	미니봉어빵	함박스테이크	치킨까스유린기	고구마맛탕/요구르트
		단무지산고추	피클단무지	반달단무지	짜사이단무지	콩나물무침
		배추김치	배추김치	배추김치	배추김치	깍두기
		765 kcal	754 kcal	871 kcal	848 kcal	955 kcal
		반반치킨	치즈돈까스&스파게티	떡자네탕앗간st떡볶이	스페셜정식	(택)돈까스김치나베
		김가루밥/우동국물	추가밥/명이미소국	김가루밥/어묵국	추가밥/유부장국	쌀밥
		설탕머니한도그	후르츠콘샐러드	비빔만두	요구르트	메추리알볼어묵조림

어드민 페이지

관리자 레스토랑 목록

새 레스토랑 추가

이름	액션
파로스 커피	수정 삭제
모야그집	수정 삭제
메가커피 가톨릭대점	수정 삭제
쇼샤돈부리	수정 삭제
나의 유부	수정 삭제
탕화쿵푸마라탕	수정 삭제
전주 막걸리 포차	수정 삭제
손가	수정 삭제
새우식탁	수정 삭제
멘야시오리	수정 삭제
학교가는길	수정 삭제
아빠짬뽕	수정 삭제
THE 53	수정 삭제
춘천애닭갈비	수정 삭제
북경	수정 삭제
고래초밥	수정 삭제
엽기떡볶이	수정 삭제
남경	수정 삭제
1983 PIZZA&PUB	수정 삭제
밀다	수정 삭제
매화스시	수정 삭제

레스토랑 수정

이름: 파로스 커피
주소: 경기 부천시 원미구 부일로6(
전화번호: 032-343-7775
평점 (개발자 평점): 4.5
메뉴:
커피

카테고리:
- [ ] 중식
- [ ] 양식
- [ ] 디저트
- [ ] 한식
- [ ] 분식
- [ ] 일식
- [x] 카페
- [ ] 술
- [ ] 고기
- [ ] 닭
- [ ] 배달
- [ ] 데이트
- [ ] 혼밥

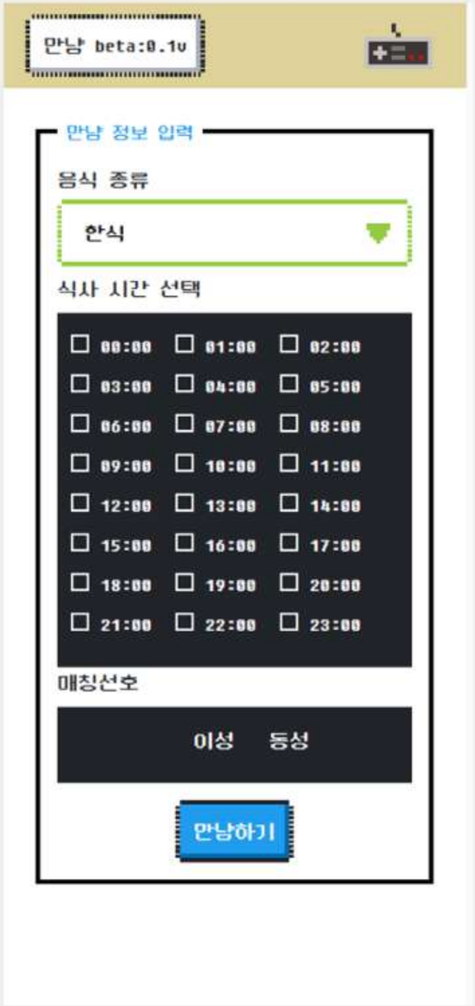
설명:
900원 커피가 가성비가 좋아요. 가대생들의 카페인을 책임지는 곳.

여행 시간 (분): 1
위도: 37.4856098029366
경도: 126.805602411375
사용자 평점: 5.0
사진 URL: 파일 선택 선택된 파일 없음
저장

초록빛	수정 삭제
이자카야 울림	수정 삭제
따붕	수정 삭제
견인지역 역곡	수정 삭제

추천 결과 수동 갱신

만냠 서비스


만냠 beta:0.1v
만냠 정보 입력
음식 종류
한식
식사 시간 선택
00:00 01:00 02:00
03:00 04:00 05:00
06:00 07:00 08:00
09:00 10:00 11:00
12:00 13:00 14:00
15:00 16:00 17:00
18:00 19:00 20:00
21:00 22:00 23:00
매칭선호
이성 동성
만냠하기

04

리뷰 기반 머신러닝 추천 기법.

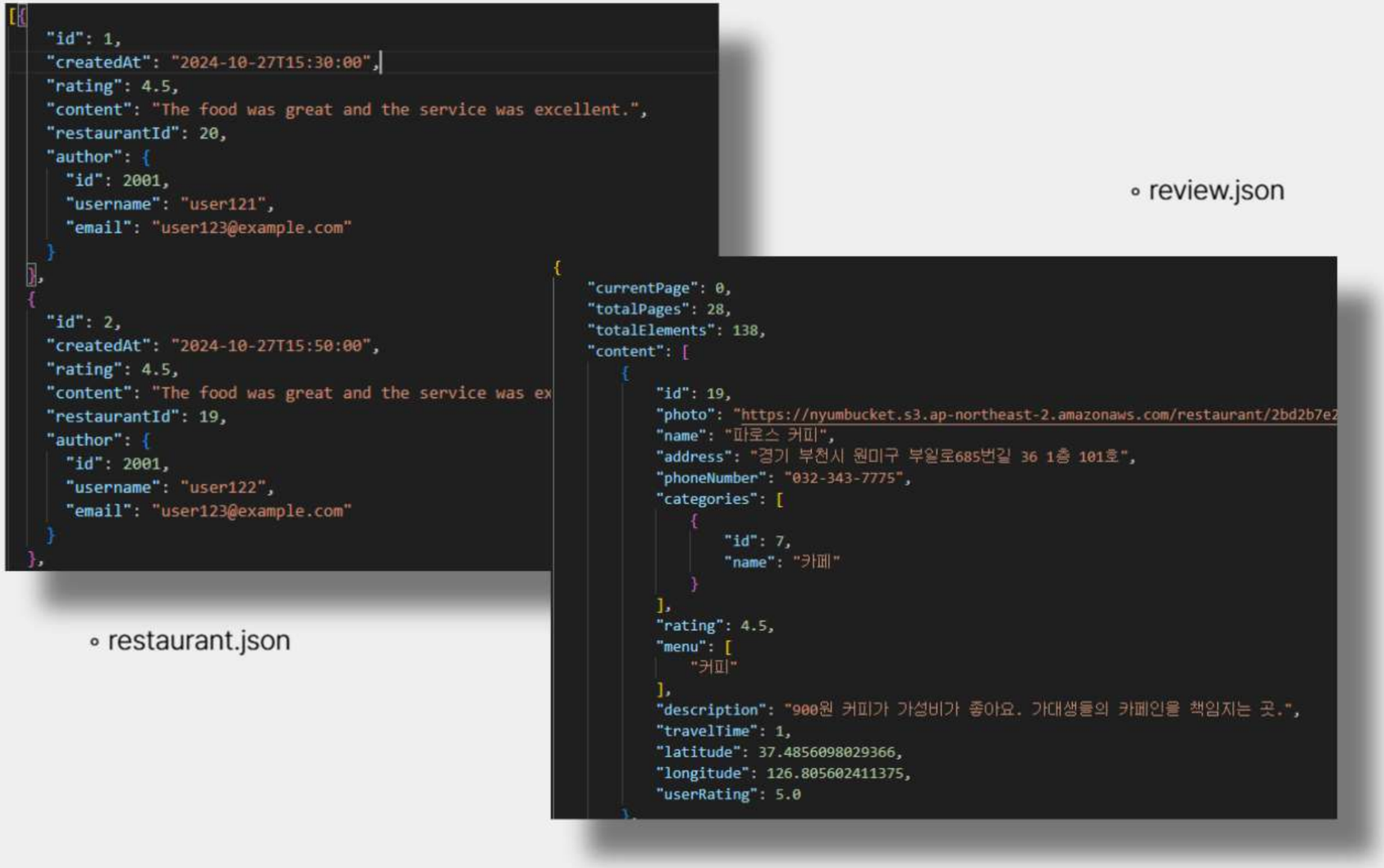
데이터를 수집하는 시스템이 수집할 수 있는 데이터의 형태,
이를 활용하는 두 가지 방식의 추천 모델

01 하이브리드 협업필터링

02 funkSVD + xgboost

03 neural_cf

DATA



```
[{
  "id": 1,
  "createdAt": "2024-10-27T15:30:00",
  "rating": 4.5,
  "content": "The food was great and the service was excellent.",
  "restaurantId": 20,
  "author": {
    "id": 2001,
    "username": "user121",
    "email": "user123@example.com"
  }
},
{
  "id": 2,
  "createdAt": "2024-10-27T15:50:00",
  "rating": 4.5,
  "content": "The food was great and the service was ex
  "restaurantId": 19,
  "author": {
    "id": 2001,
    "username": "user122",
    "email": "user123@example.com"
  }
},
```

◦ restaurant.json

◦ review.json

```
{
    "currentPage": 0,
    "totalPages": 28,
    "totalElements": 138,
    "content": [
        {
            "id": 19,
            "photo": "https://nyumbucket.s3.ap-northeast-2.amazonaws.com/restaurant/2bd2b7e2
            "name": "파로스 커피",
            "address": "경기 부천시 원미구 부일로685번길 36 1층 101호",
            "phoneNumber": "032-343-7775",
            "categories": [
                {
                    "id": 7,
                    "name": "카페"
                }
            ],
            "rating": 4.5,
            "menu": [
                "커피"
            ],
            "description": "900원 커피가 가성비가 좋아요. 가대생들의 카페인을 책임지는 곳.",
            "travelTime": 1,
            "latitude": 37.4856098029366,
            "longitude": 126.805602411375,
            "userRating": 5.0
        },
```

01
하이브리드 협업필터링

아이템 기반 협업필터링

- 아이템-아이템 유사도
- 한 유저에 대해서만 예측

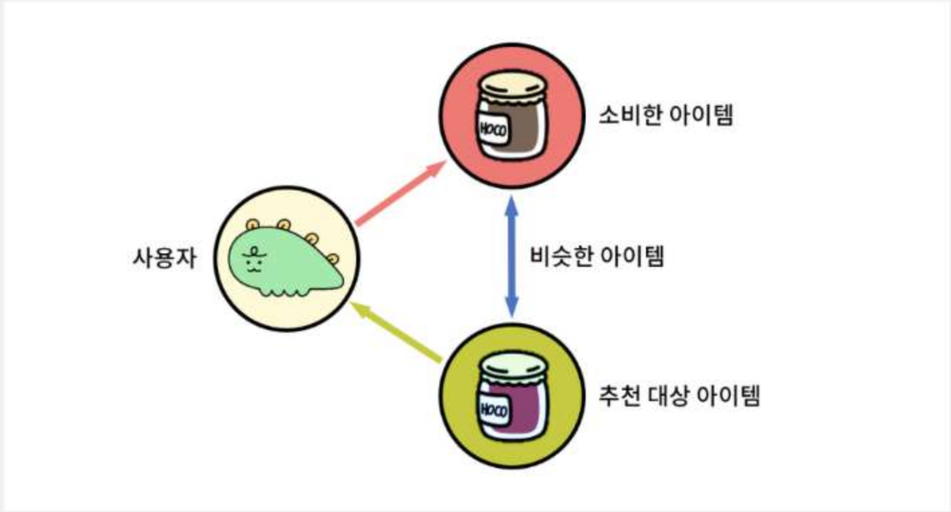
하이브리드 협업필터링

- 아이템 및 사용자 앙상블
- cold start 완화

사용자 기반 협업필터링

- 사용자-사용자 유사도
- 한 가게에 대해서만 예측

01-1
아이템 기반 협업필터링



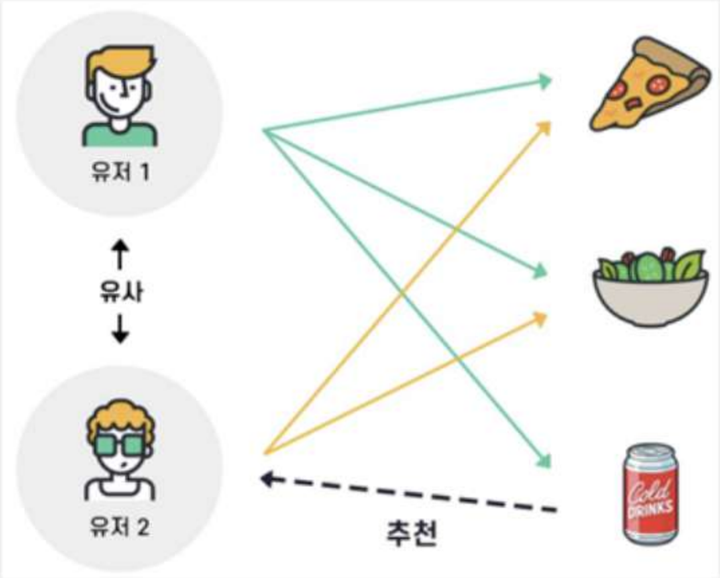
https://tech.kakao.com/posts/486

01 사용자가 소비한 아이템에 대한 정보

02 아이템간의 유사도 계산

03 유사한 아이템 사용자에게 추천

01-2
사용자 기반 협업필터링



https://brunch.co.kr/@biginsight/15

01 사용자가 소비한 아이템에 대한 정보

02 사용자간의 유사도 계산

03 유사한 사용자가 소비한 아이템 추천

01-3
유사도 계산

$$\text{similarity} = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}$$

코사인 유사도

https://wikidocs.net/24603

01-3
유사도 계산

사용자 간 유사도 행렬:

	2001	2002	2003	2004
2001	1.000000	-0.647339	-0.584031	-0.307403
2002	-0.647339	1.000000	0.911783	0.495143
2003	-0.584031	0.911783	1.000000	0.808259
2004	-0.307403	0.495143	0.808259	1.000000

	19	14	20	12	26
19	1.000000	-0.039185	-0.470012	0.015094	-0.532801
14	-0.039185	1.000000	-0.382656	-0.132695	0.092093
20	-0.470012	-0.382656	1.000000	0.147399	0.360158
12	0.015094	-0.132695	0.147399	1.000000	-0.035474
26	-0.532801	0.092093	0.360158	-0.035474	1.000000

사용자

'user_total_rating','user_total_count',
'유저_다른_레스토랑_평균', '유저_레스토랑_최고_평점',
'유저_레스토랑_최근_리뷰_경과시간(일)', '유저_레스토랑_리뷰_개수','리뷰 내용'

아이템

'category1','category2', 'content1','content2'
'menu_text, 'text_features',
'rating', 'userRating'

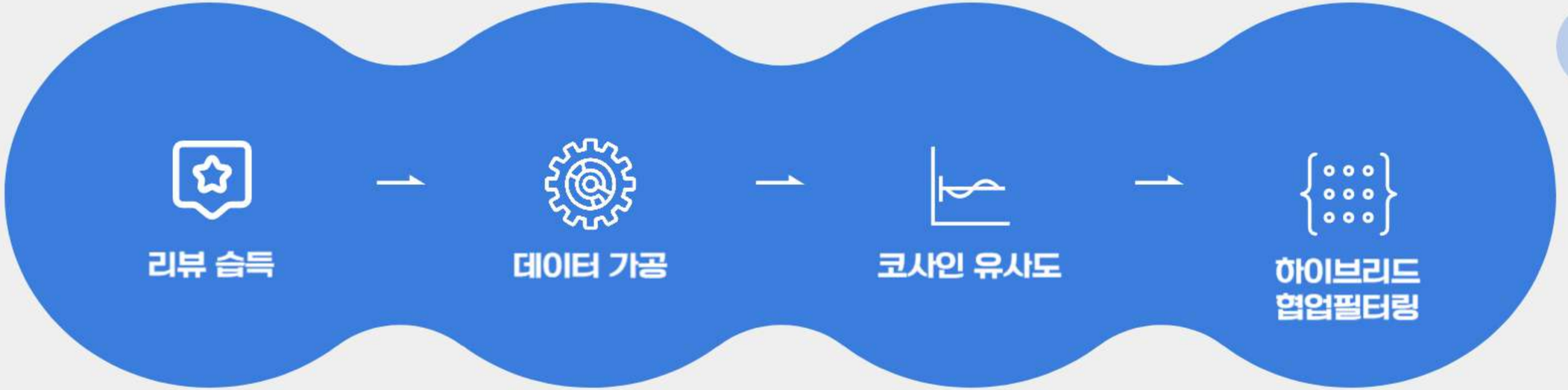
01-4
예측행렬

	19	14	20	12	26
2001	4.5	4.5	4.5	4.5	4.5
2002	3.0	1.5	1.5	1.5	1.5
2003	3.5	2.5	2.5	2.5	2.5
2004	2.25	0.0	0.692796	0.0	0.0

```python
# 수정된 하이브리드 예측 함수
def hybrid_predict(predicted_df_user_based, predicted_df_item_based):
    # 우선 NaN이 아닌 값을 우선적으로 사용하여 병합
    hybrid_pred = predicted_df_user_based.combine_first(predicted_df_item_based)

    # 둘 다 값이 존재하는 경우 평균으로 덮어쓰도록 설정
    both_non_nan = predicted_df_user_based.notna() & predicted_df_item_based.notna()
    hybrid_pred[both_non_nan] = (predicted_df_user_based[both_non_nan] + predicted_df_it

    return hybrid_pred

# 4. 하이브리드 예측 평점 계산
hybrid_predicted_df = hybrid_predict(predicted_df_user_based, predicted_df_item_based)

# 결과 출력
print(hybrid_predicted_df)
```

01-5

모델 1 pipeline



- 유저로부터 리뷰 데이터 습득
- 습득 시간대 별 저장

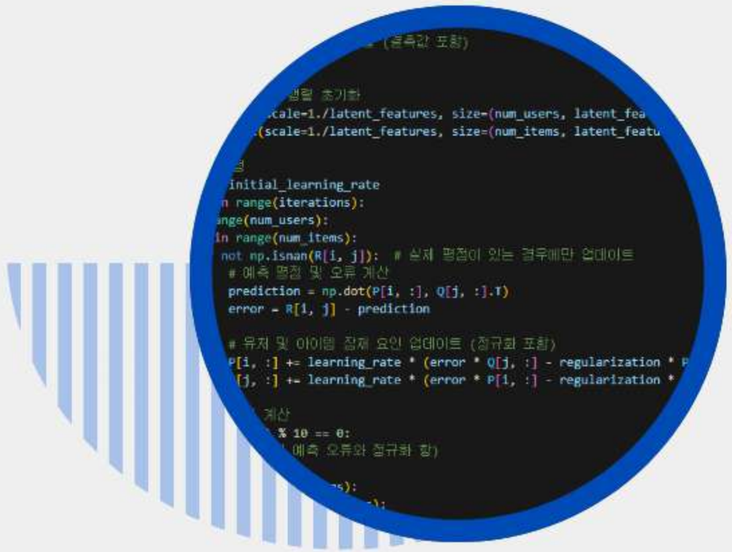
- 가게 정보 호출 api
- 필터링 방법별 데이터 셋 생성

- 아이템-아이템 유사도
- 유저-유저 유사도

- 유사도를 통한 필터링
- 각각의 필터링 앙상블

02

funkSVD + xgboost



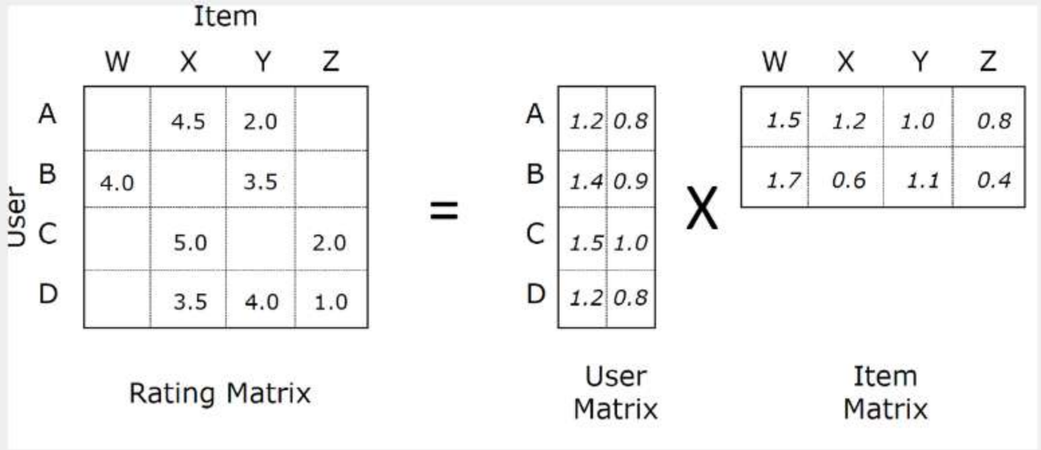
funkSVD

svd가 가지는 문제점을 해결
모든 유저-모든 음식점간의 평가 점수 예측

XGBRegressor

서버의 과부하로 인한 언더피팅 문제 방지
svd에 주어지지 않은 특성에 대한 정보 학습

02-1
matrix factorization



https://supkoon.tistory.com/31

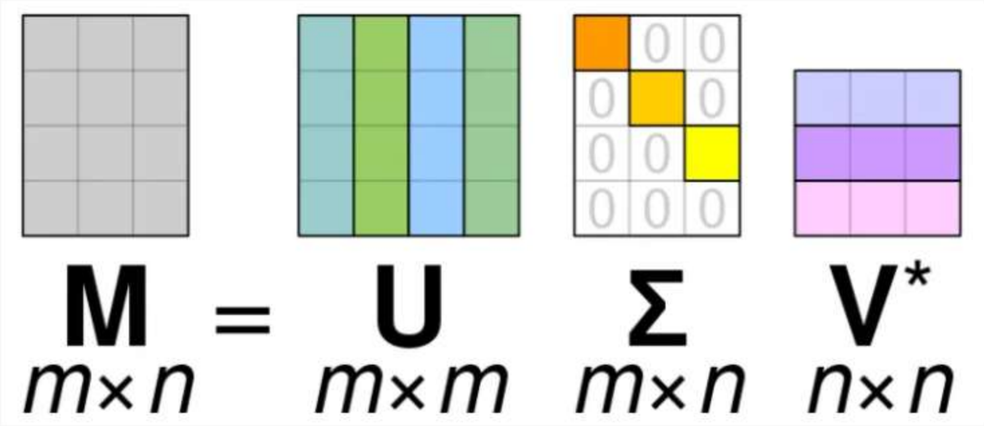
02-2
matrix factorization

```
Iteration: 10, Loss: 48.8638
Iteration: 20, Loss: 48.2878
Iteration: 30, Loss: 46.2686
Iteration: 40, Loss: 39.8269
Iteration: 50, Loss: 25.1847
Iteration: 60, Loss: 9.4049
Iteration: 70, Loss: 3.2460
Iteration: 80, Loss: 1.9871
Iteration: 90, Loss: 1.6784
Iteration: 100, Loss: 1.5650
            19        14        20        12        26
2001  4.314974 -0.035376  4.475921 -0.000149  0.041042
2002  1.223669 -0.009266  1.323913 -0.001669  0.010904
2003  2.048223 -0.020503  2.212613 -0.003268  0.018667
2004  0.006930  0.000545  0.002651  0.000200 -0.000150
2005 -0.003299 -0.000345 -0.004147 -0.000680  0.000879
2006 -0.014034  0.000305 -0.017789 -0.000438 -0.001244
2007  0.006469  0.000132  0.002881  0.000834  0.000408
```

```python
# 설정: 잠재 요인 수, 학습률, 정규화 파라미터, 반복 횟수
latent_features = 100
learning_rate = 0.01
regularization = 0.1
iterations = 100

# 데이터 준비: 결측값을 0으로 대체하고 MF 적용
R = df.fillna(0).values
num_users, num_items = R.shape

# 유저 및 아이템 잠재 요인 행렬 초기화
P = np.random.normal(scale=1./latent_features, size=(num_users, latent_features))
Q = np.random.normal(scale=1./latent_features, size=(num_items, latent_features))

# 손실 함수를 계산하는 함수
def calculate_loss(R, P, Q, non_zero_indices, regularization):
    loss = 0
    for i, j in non_zero_indices:
        # 현재 평점과 예측 평점의 차이 계산
        error = R[i, j] - np.dot(P[i, :], Q[j, :].T)
        loss += error ** 2 + regularization * (np.linalg.norm(P[i, :]) + np.linalg
    return loss

# 실제 평점이 있는 인덱스 추출 (NaN이 아닌 값)
non_zero_indices = [(i, j) for i in range(num_users) for j in range(num_items) if
```

02-3

Singular Value Decomposition(SVD)

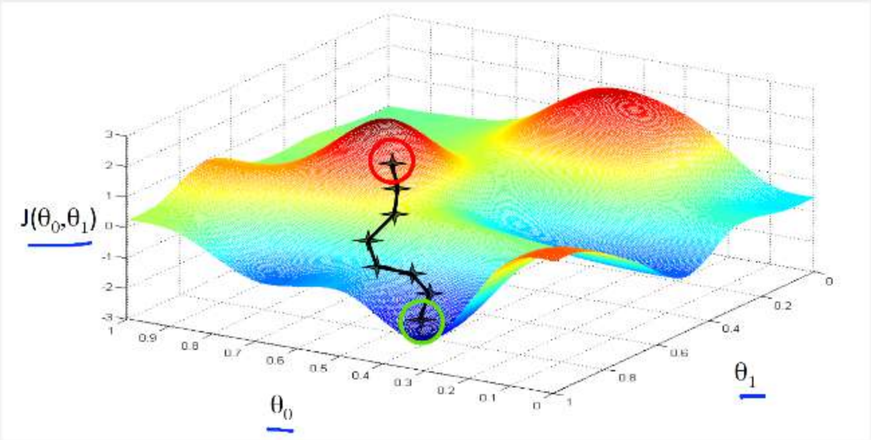


https://ko.wikipedia.org/wiki/%ED%8A%B9%EC%9E%87%EA%B0%92_%EB%B6%84%ED%95%B4

movieId	1	2	3	4	5	6	7	8	9	10
userId										
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

svd 의 장점: sigma에서 모든 feature를 사용할 필요 없이 가장 영향도가 큰 k개의 feature만 사용해도 유사한 원본을 얻을 수 있음
svd 의 문제점 : 결측치에 대한 처리가 어려움

02-4
funkSVD



https://jiwonkkim.github.io/machine%20learning/2021/01/09/ml_gradient_descent/

01 Feature를 한번에 하나씩만 학습

02 확률적 경사하강법(SGD) 사용

03 유저가 아직 평가하지 않은 값을 처리

02-4
funkSVD

	19	14	20	12	26
2001	4.270059	0.126212	3.052739	-0.005500	-1.153681
2002	1.967896	0.073238	2.795515	0.056215	-0.683771
2003	1.946591	0.094382	2.822865	0.022566	-0.903070
2004	-1.101550	-0.022552	-1.061986	-0.081136	2.349085
2005	-0.221106	-0.010409	-0.314210	-0.018340	0.205891
2006	0.008846	-0.007631	0.023714	0.001522	0.123680
2007	0.470013	0.007822	0.351278	-0.003303	-0.370333

funkSVD로 예측된 유저 리뷰 점수

```
latent_features = 10
initial_learning_rate = 0.01
regularization = 0.1
iterations = 10000

# 유저-아이템 평점 행렬 R 생성
R = df.values  # 유저-아이템 평점 행렬 (결측값 포함)
num_users, num_items = R.shape

# 유저와 아이템 잠재 요인 행렬 초기화
P = np.random.normal(scale=1./latent_features, size=(num_users, latent_features))
Q = np.random.normal(scale=1./latent_features, size=(num_items, latent_features))

# FunkSVD 학습 과정
learning_rate = initial_learning_rate
for iteration in range(iterations):
    for i in range(num_users):
        for j in range(num_items):
            if not np.isnan(R[i, j]):  # 실제 평점이 있는 경우에만 업데이트
                # 예측 평점 및 오류 계산
                prediction = np.dot(P[i, :], Q[j, :].T)
                error = R[i, j] - prediction
```

02-5
XGBOOST

```
# DataFrame을 float로 변환하여 결측값 포함 최종 예측 평점 행렬 출력
final_predicted_df = final_predicted_df.astype(float)

print("추천 결과 계산이 완료되었습니다.")

@app.route('/recommendations', methods=['GET'])
def get_recommendations():
    recommendations_json = final_predicted_df.to_json(orient='index')
    return jsonify(json.loads(recommendations_json))

if __name__ == '__main__':
    app.run(debug=True, port=5000)
```

```
# 3. 학습을 위한 특성(X)과 타겟(y) 정의
X = train_df[['svd_pred', 'userRating', 'travelTime', 'rating', 'user_total_rating', 'user_total_count',
              'user_restaurant_rating', 'user_restaurant_count', '유저_니른_레스토랑_평균',
              '유저_레스토랑_최고_평점', '유저_레스토랑_최근_리뷰_경과시간(일)']].fillna(0)
y = train_df['reviewRating']

# 4. XGBoost 모델 학습
xgb_model = XGBRegressor(objective='reg:squarederror', n_estimators=100, learning_rate=0.1,
                         max_depth=5, random_state=42)
xgb_model.fit(X, y)
```

funkSVD 에서 적용되지 않은 특성에 대해 추가적인 학습과정을 거쳐 단순히 점수만이 아닌 다른 여러 연관 데이터의 반영이 가능

funkSVD에서 충분한 학습이 이루어지지 않았다 하더라도 해당 과정을 통해 피드백이 가능

결측치 없는 추천점수 데이터의 도출로 사용자에게 맞춤화 추천이 구현

02-6

모델 2 pipeline

- 유저로부터 리뷰 데이터 습득
- 습득 시간대 별 저장

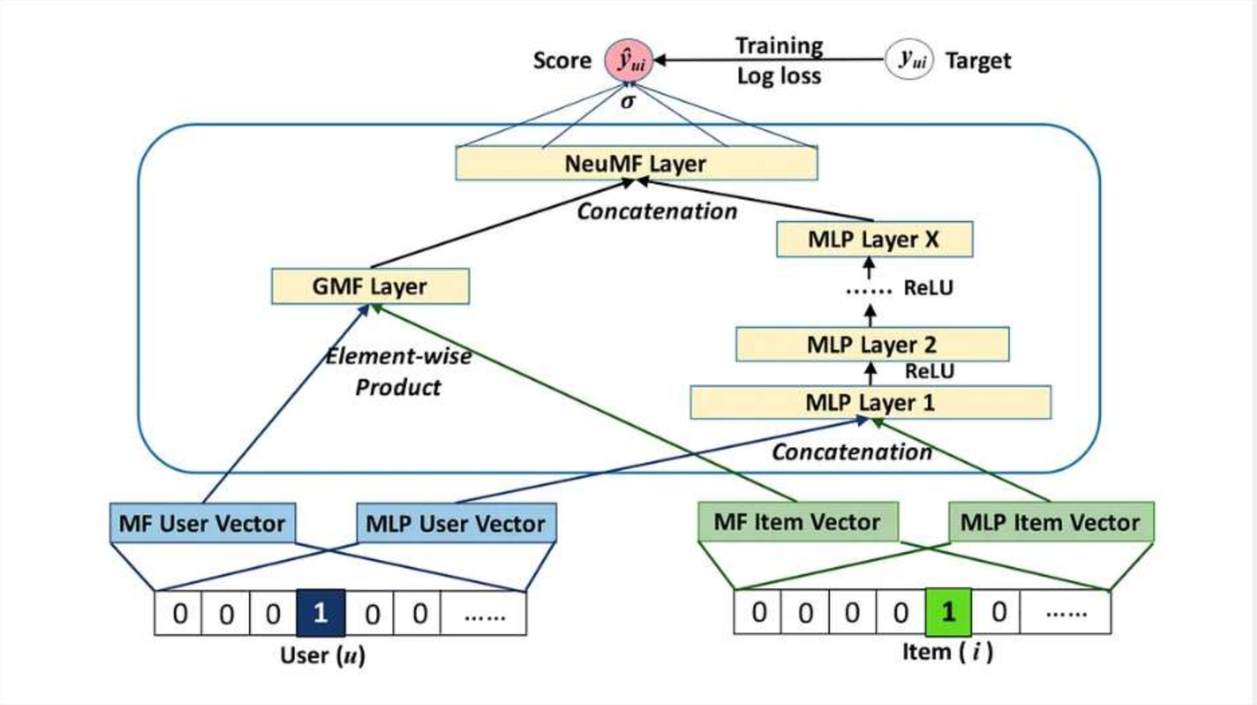
- 가게 정보 호출 api
- 20개 이상의 피처로 재구성

- 경사하강법(SVD)
- 모든 유저,가게 점수 예측

- 추가적인 특성 학습
- 언더피팅 방지, 서버과부하 방지

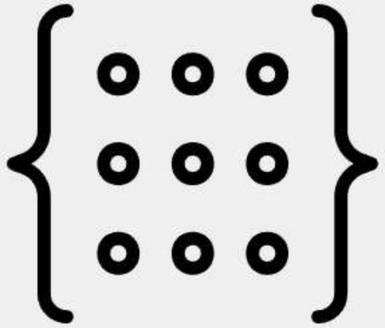
03

neural_cf

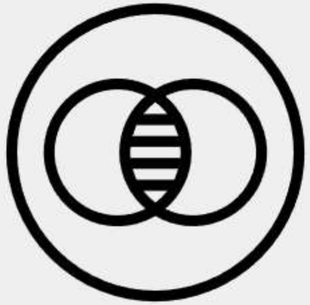

Score
$\hat{y}_{ui}$
Training
Log loss
y_{ui}
Target
σ
NeuMF Layer
Concatenation
MLP Layer X
GMF Layer
..... ReLU
MLP Layer 2
Element-wise
Product
ReLU
MLP Layer 1
Concatenation
MF User Vector
MLP User Vector
MF Item Vector
MLP Item Vector
0 0 0 1 0 0 .....
User (u)
0 0 0 0 1 0 .....
Item (i)

03-1

collaborated mf



Generalized Matrix Factorization(GMF) Layer
일반화 행렬 분해방법. 사용자-아이템 상호작용의 선형구조



Fusion of GMF and MLP Layer
GMF와 MLP의 병합.

Multi-Layer Perceptron(MLP) Layer
다층 퍼셉트론. 비선형구조의 학습

03-2

neural_cf

user_embedding과 item_embedding을 통해
GMF와 유사한 역할을 수행

사용자, 아이템, 그리고 추가적인 특성의 임베딩들을 모두 결합하여
MLP 계층에 입력으로 사용하고, [128, 64, 32, 16] 크기의
은닉 레이어를 통해 복잡한 패턴을 학습

각 MLP 레이어는 ReLU 활성화를 통해 비선형 변환을 수행하며,
이를 통해 단순한 GMF로는 포착할 수 없는 복잡한 비선형 관계를 학습

```python
# NCF Model with additional features
class NCF(nn.Module):
    def __init__(self, num_users, num_items, embedding_dim, mlp_hidden_layers,
                 num_cat_features, num_cat_unique_values, num_num_features):
        super(NCF, self).__init__()
        # Embeddings for users and items
        self.user_embedding = nn.Embedding(num_users, embedding_dim)
        self.item_embedding = nn.Embedding(num_items, embedding_dim)

        # Embeddings for categorical features
        self.cat_embedding_layers = nn.ModuleList([
            nn.Embedding(num_unique, embedding_dim) for num_unique in num_cat_unique_values
        ])

        # Linear layer for numerical features
        self.num_linear = nn.Linear(num_num_features, embedding_dim)

        # MLP layers
        mlp_input_size = embedding_dim * (2 + len(num_cat_unique_values) + 1)  # user, item, c
        mlp_layers = []
        for layer_size in mlp_hidden_layers:
            mlp_layers.append(nn.Linear(mlp_input_size, layer_size))
            mlp_layers.append(nn.ReLU())
            mlp_input_size = layer_size
        self.mlp = nn.Sequential(*mlp_layers)
```

03-3

모델 3 pipeline

- 유저로부터 리뷰 데이터 습득
- 습득 시간대 별 저장

- GMF로 선형구조 학습
- 사용자, 아이템, 범주형, 수치형 임베딩을 결합

- 강력한 표현력을 갖춘 벡터가 생성
- MLP로 비선형구조 학습

- 최종 예측 레이어
- 단일 스칼라 값(예측 평점)

05

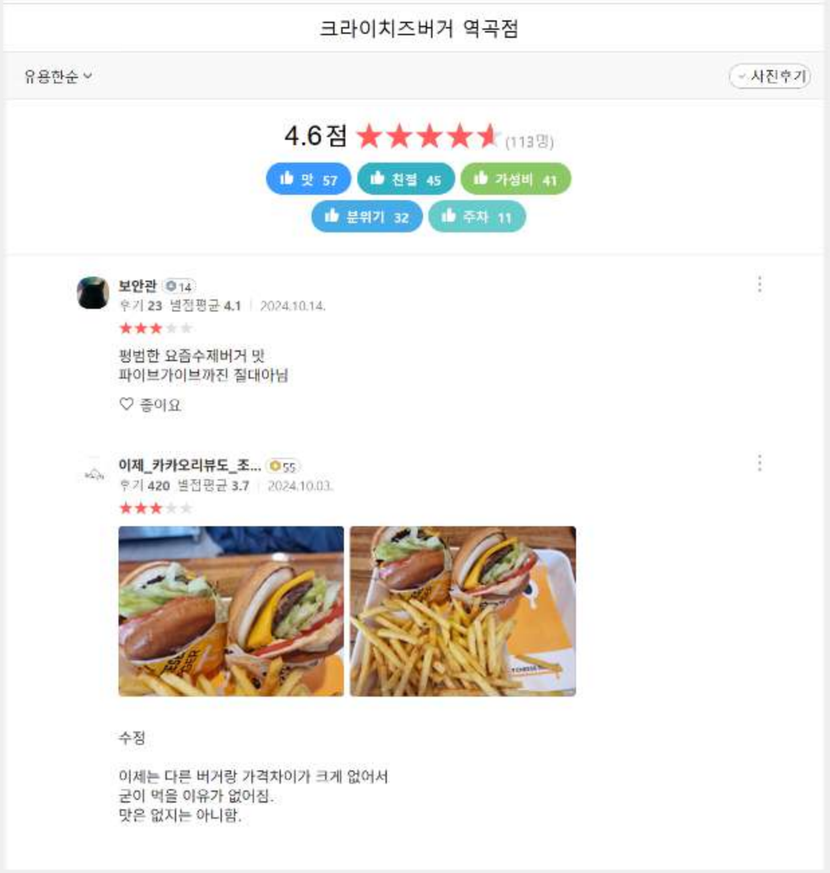
사용자 및 시장성.

모델의 간단한 성능을 확인하고
냠톨릭의 기존 사용자층과 활동 데이터를 살펴 봅니다.

01 모델의 임시 테스트

02 사용자 활동 및 데이터

임시 데이터셋



kakao맵 후기서비스를 통해 데이터 셋을 생성
총 12개의 실제 사용자 후기(7명의 사용자, 3개의 가게)를 추출
이 중 한 유저에 대해 주목하여 가게들에 대한 후기 점수를 예측

임시 테스트

	1	2	3
2001	3.487838	3.582812	3.675187
2002	3.116429	3.224333	3.301110
2003	3.662305	3.808831	3.907660
2004	3.899577	4.021204	4.122964
2006	3.673032	3.820996	3.942516
2007	3.882491	4.030188	4.171573

해당 모델의 검증을 위해 한 유저가 후기를 남긴 가게들에 대해 후기점수 예측을 진행

1.0~5.0까지의 범위를 통해 NMAE 계산를 계산하여 실제 음식점에 대해 정확도를 계산

Accuracy: 91.34%

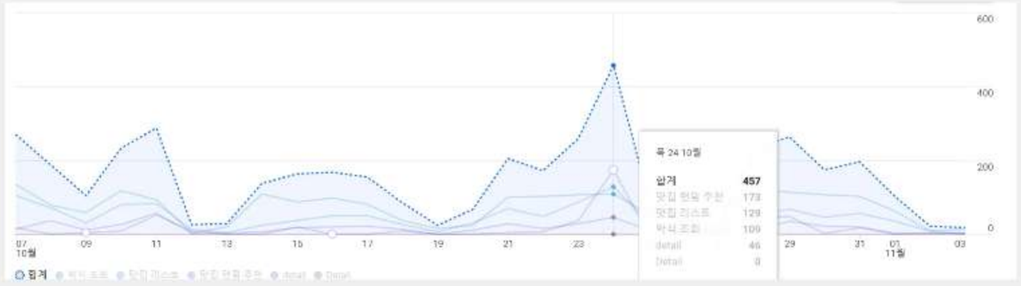
사용자 활동 및 데이터


이벤트 수
7.4만
주요 이벤트
0
조회수
2.6만
활성 사용자
4.4천
2천
1.5천
1천
500
0
01
1월
01
4월
01
7월
01
10월
지난 12개월
이전 기간
지난 12개월
보고서 스냅샷 보기


활성 사용자
683
71.3%
이벤트 수
1.2만
58.3%
주요 이벤트
0
총수익
$0.00
1.5천
1천
500
0
06
10월
13
20
27
03
지난 30일
이전 기간
지난 30일
수 23 10월 vs 수 25 9월
활성 사용자
54
54.6%

사용자 활동 및 데이터

	페이지 제목 및 화면 클래스	↓ 조회수	활성 사용자	활성 사용자당 조회수	활성 사용자당 평균 참여 시간	이벤트 수 모든 이벤트	주요 이벤트 모든 이벤트	총수익
	합계	4,199 총계 대비 100%	662 총계 대비 100%	6.34 평균과 동일	43초 평균과 동일	11,352 총계 대비 100%	0.00	$0.00
1	학식 조회	1,985	419	4.74	23초	4,927	0.00	$0.00
2	맛집 리스트	1,271	357	3.56	39초	3,407	0.00	$0.00
3	맛집 퀀칠 추천	477	43	11.09	35초	1,772	0.00	$0.00
4	detail	464	156	2.97	20초	1,238	0.00	$0.00
5	Detail	2	2	1.00	5초	7	0.00	$0.00
6	로컬리스트	0	1	0.00	1초	1	0.00	$0.00



꾸준한 사용자가 존재함을 수치를 통해, 트래픽을 통해 확인할 수 있습니다. 구체적인 홍보없이 꾸준히 사용자가 늘고 있음을 보여주며 시장성이 있음을 직접적으로 확인할 수 있습니다.

06

발전가능성.

냠톨릭2.0은 자체적으로 데이터를 수집할 수 있습니다.
현재 개발된 모델들에 대해 테스트가 가능하며 딥러닝, 시계열로 확장을 준비합니다.

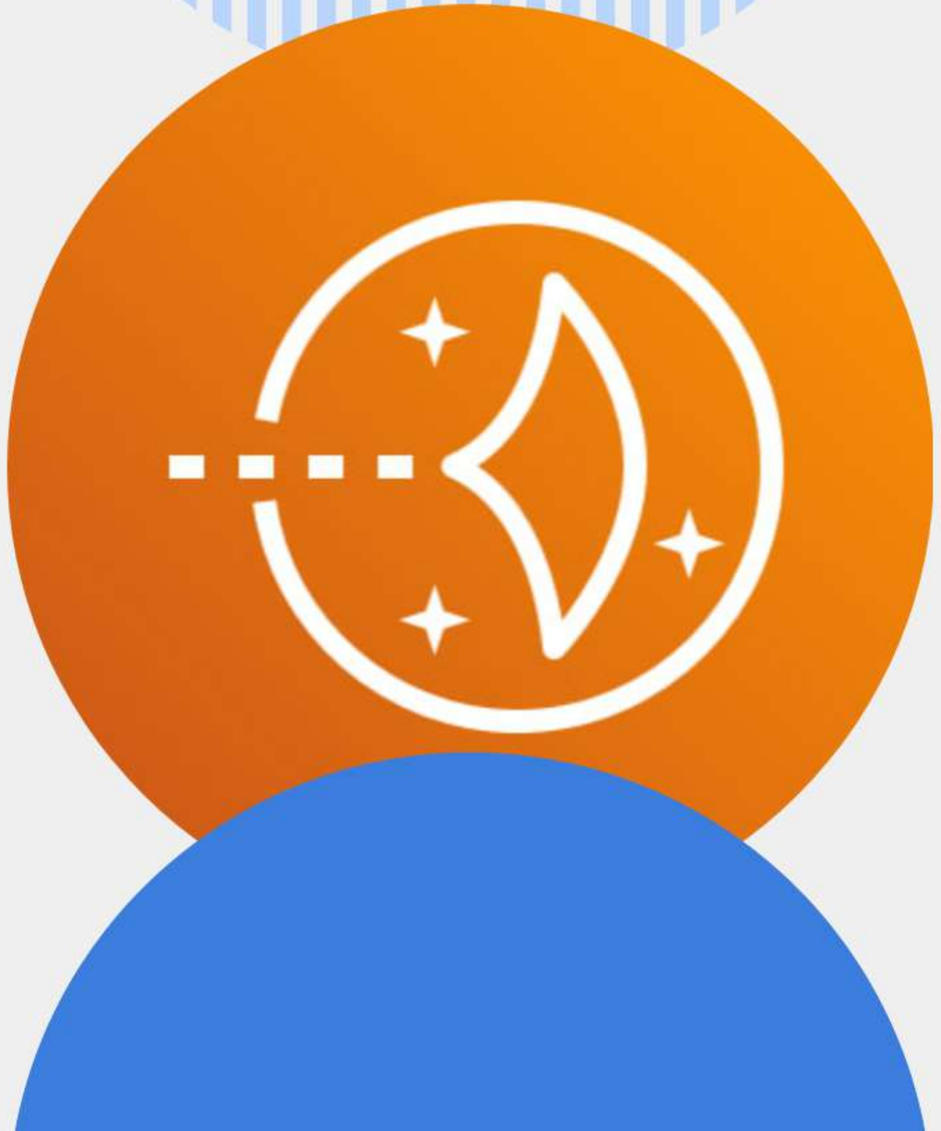
01 모델의 성능 테스트 계획

02 서버의 확장과 적용해볼 모델

DATA 저장
냠톨릭2.0

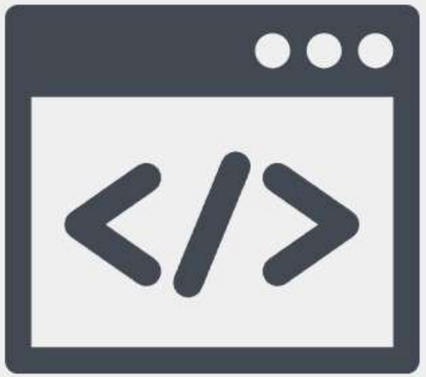
amazon lightsail을 이용한 유저 데이터의 저장

냠톨릭은 온프레미스가 아닌 클라우드 시스템 상에서 작동합니다.
전세계 최대 클라우드 서비스인 amazon web service를 사용해서 구현하고 저장하며 자제척인 이원화, 가용성을 통해 데이터의 분실에 대한 가능성을 최소로 유지합니다.
시간대별 모든 데이터를 기록할 수 있기에 추천 모델을 통해 나온 결과값의 정확도를 계산할 수 있습니다.



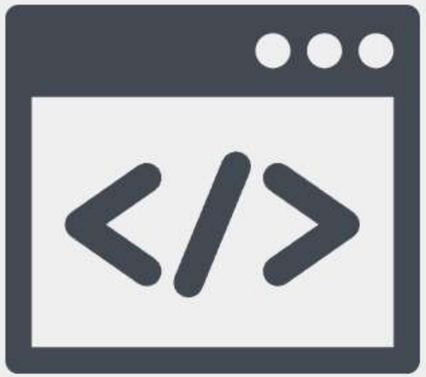
01

성능테스트



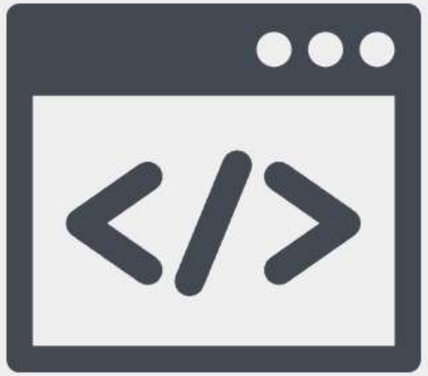
하이브리드 협업필터링
=item based+user based

VS



funkSVD
=mf+svd+funkSVD+all xgboost

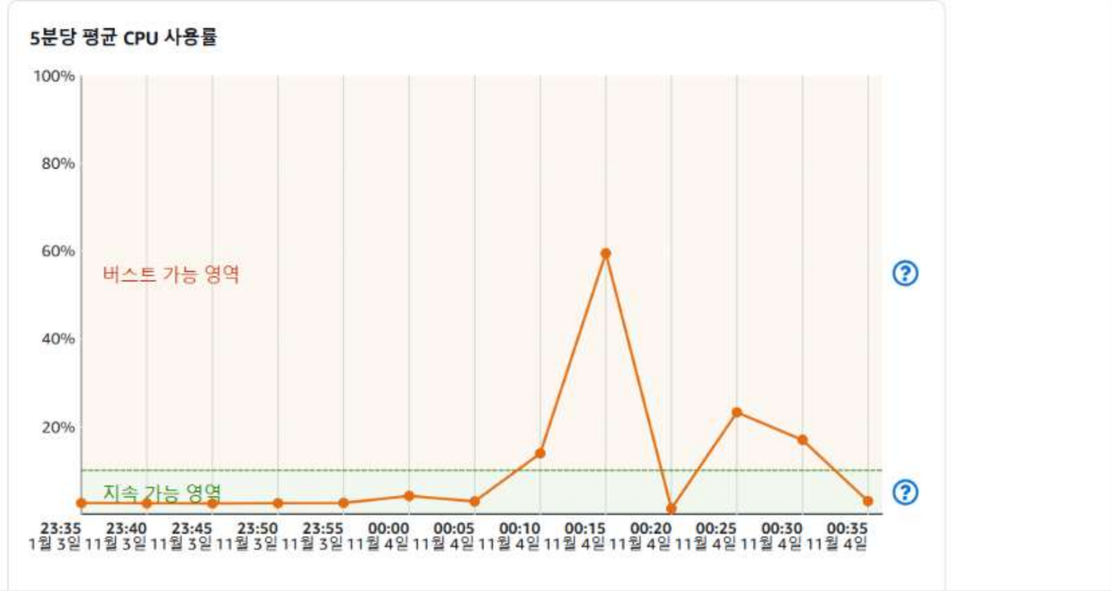
VS



neural_cf

02

추가 개발계획



딥러닝 모델이나 기타 모델들의 경우 상당한 계산량이 필요하기에
높은 수준의 cpu, gpu 성능과 사용량이 필요시됩니다.

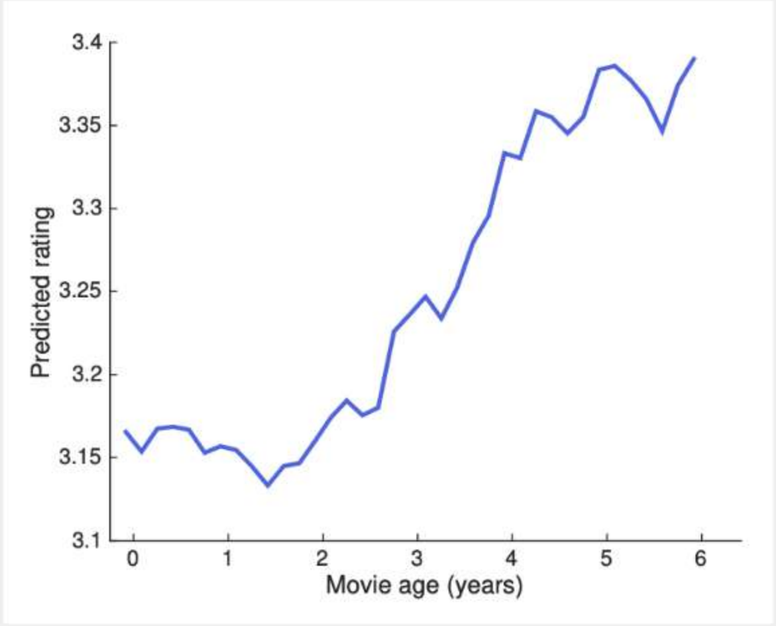
이를 위한 확장된 서버의 경우 현재 유지중인 서버(5,000원/월)
수준에 비해 비용부담이 발생합니다.

이에 상금이나 광고 수익을 통해 자본이 확보된다면 추가적인 딥러닝,
시계열기반의 모델을 사용하는 것을 고려할 수 있습니다.

02
추가 개발계획

Google Maps

구글 맵 데이터 크롤링을 비롯하여 외부 데이터 셋 생성 후,
해당 서비스의 데이터 파인튜닝



시계열 요소를 포함한 Matrix Factorization 모델 (TimeSVD++ 등)
->계절별 선호도 변화 파악가능

참고자료

https://link.springer.com/chapter/10.1007/978-3-031-61221-3_12

https://dl.acm.org/doi/abs/10.1145/2043932.2043988

https://dl.acm.org/doi/abs/10.1145/358916.358995

https://dl.acm.org/doi/abs/10.1145/371920.372071

https://tech.kakao.com/posts/463

https://www.rdocumentation.org/packages/recommenderlab/versions/1.0.6/topics/funkSVD

https://heegyukim.medium.com

https://dl.acm.org/doi/pdf/10.1145/3018661.3018689

https://github.com/rudolfsteiner/CollaborativeFilteringV3

감사합니다.

THANK YOU!

numtolic2.0