특화 프로젝트

빅데이터

맛집 추천 시스템

A405

구영지 이용석 이재욱 한우석 홍지희

2021.03.25

목차

01

프로젝트 리마인드

- 빅데이터란?
- 프로젝트 소개
- 프로젝트 개요

빅데이터 이해하기

빅데이터

: 대규모의 데이터. 데이터로부터 가치를 추출하고 결과를 분석하는 기술

amazon

경쟁 업체 분석 / 이익 창출

상권 분석 / 입점 위치 탐색

프로젝트 주제

사용자 리뷰 기반의
맛집 추천 서비스

프로젝트 개요

데이터 분석 및 정리

데이터 가공 및 저장

사용자-아이템 행렬 생성

기본 추천 시스템 구축

협업 필터링

컨텐츠 기반 필터링

서비스 고도화

하이브리드 추천 시스템 구축

성능측정 모니터링 시스템

잠시 다음 챕터 들어가기 전에...

02 유사도 계산

- 추천시스템 유사도 계산
- 코사인 유사도
- 피어슨 유사도

프로젝트 개요

유사도(Similarity) 계산

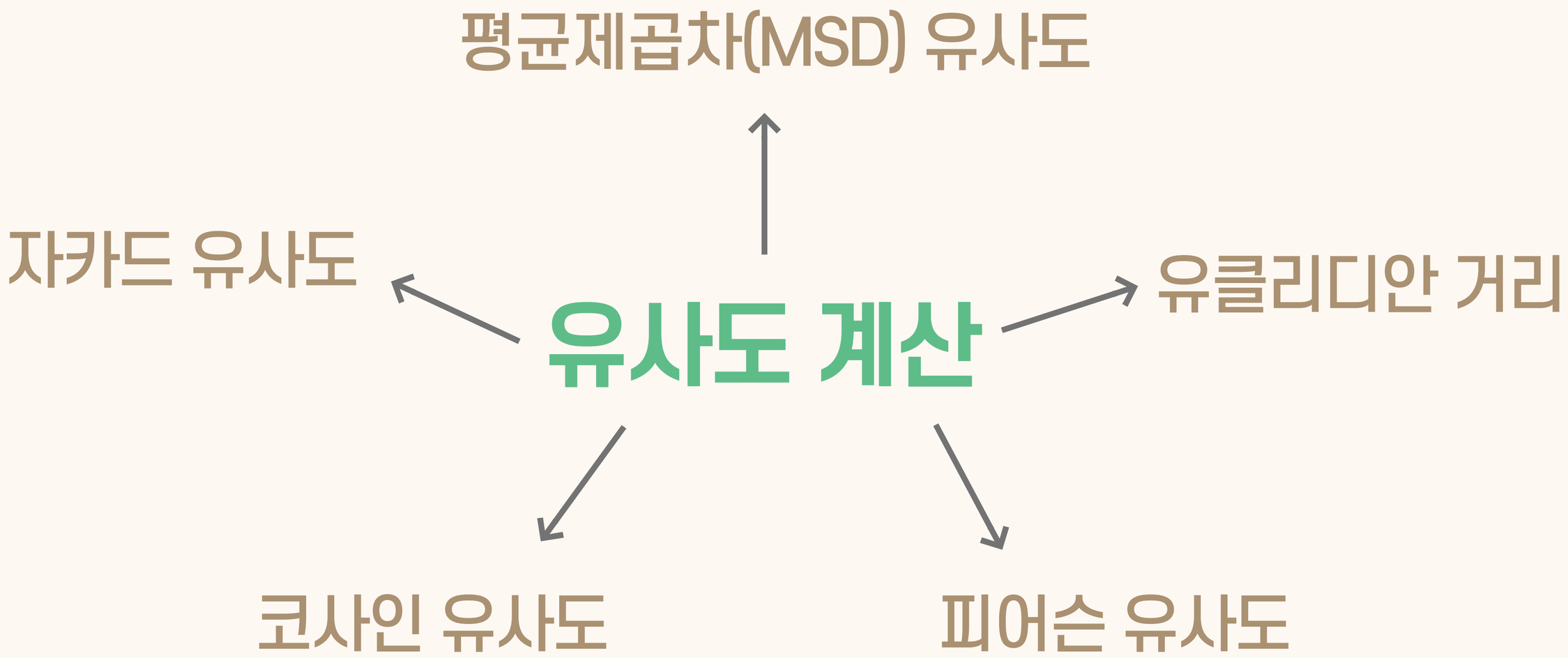
: 두 데이터가 얼마나 같을지 나타내는 척도

ex) 표절 검사

추천 시스템에서 유사도 계산

: 사용자에게 무언가를 추천하기 위한 아이템을 찾기 위해서는 유사도 계산이 필수적

추천시스템 유사도 계산



코사인 유사도

$$\text{cosine\_sim}(u, v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

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

분배법칙

$$|\vec{a} - \vec{b}|^2 = |\vec{a}|^2 + |\vec{b}|^2 - 2\vec{a} \cdot \vec{b}$$

제2코사인법칙

$$|\vec{a} - \vec{b}|^2 = |\vec{a}|^2 + |\vec{b}|^2 - 2|\vec{a}||\vec{b}| \cos\theta$$

즉 $\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta$

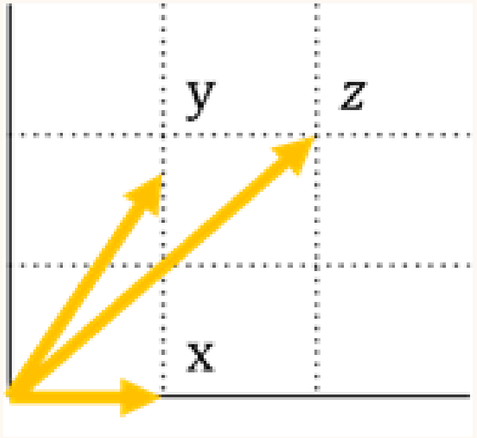
- 코사인 값의 범위 : -1 ~ 1

- 코사인 유사도 값이 가지는 의미

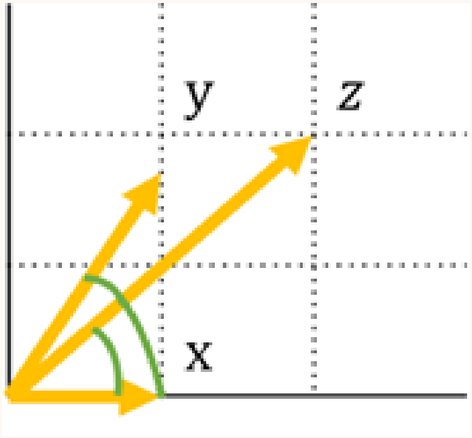
  -1 : 반대 성향이다
  0 : 서로 독립적이다
  1 : 완전히 같다

-> 즉, 1에 가까울수록 유사하다.

코사인 유사도



< x,y,z 좌표 >



< x에 대한 y,z의 각도 >

In [8]:
```
import math
import numpy as np

def cosine_sim(a, b):
    sum_a = 0
    sum_b = 0
    sum_a_b = 0
    length = len(a)
    for i in range(length):
        sum_a += pow(a[i], 2)
        sum_b += pow(b[i], 2)
        sum_a_b += a[i]*b[i]

    return sum_a_b / (math.sqrt(sum_a)*math.sqrt(sum_b))
```

In [9]:
```
x = np.array([2,0])
y = np.array([2,3])
z = np.array([4,4])
```

In [12]:
```
cos_x_y = cosine_sim(x,y)
cos_x_z = cosine_sim(x,z)
print('x,y의 코사인 유사도 : ' + str(cos_x_y))
print('x,z의 코사인 유사도 : ' + str(cos_x_z))
```

```
x,y의 코사인 유사도 : 0.5547001962252291
x,z의 코사인 유사도 : 0.7071067811865475
```

< 예시에 대해 코사인 유사도 계산한 코드 >

피어슨 유사도

$$\text{pearson\_sim}(i,j) = \frac{\sum_{u \in U_{ij}} (r_{ui} - \mu_i) \cdot (r_{uj} - \mu_j)}{\sqrt{\sum_{u \in U_{ij}} (r_{ui} - \mu_i)^2} \cdot \sqrt{\sum_{u \in U_{ij}} (r_{uj} - \mu_j)^2}}$$

$$\text{cosine\_sim}(u,v) = \frac{\sum_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum_{i \in I_{uv}} r_{vi}^2}}$$

	user1	user2	user3	user4	user5
item1	1	2.4	1.3333	-1.25	-1
item2	0	-1.6		1.75	
item3	1	0.4		-2.25	1
item4		0.4	0.3333		-1
itme5			-1.6667	1.75	1
item6	-2	-1.6			0

	user1	user2	user3	user4	user5
item1	4	4	4	1	1
item2	3	0		4	
item3	4	2		0	3
item4		2	3		1
itme5			1	4	3
item6	1	0			2
유저별 평균	3	1.6	2.6667	2.25	2

03 컨텐츠 기반 필터링

- TF-IDF

	특징1	특징2	특징3	특징4	특징5	특징6
아이템1						
아이템2						
아이템3						
아이템4						
아이템5						
아이템6						

컨텐츠에 대한 **특징**을
어떻게 정할 것 인가?

[아이템 x 특징]으로 구성된 행렬에서 **유사도 계산**을 이용하여
유사한 아이템을 선택하여 추천해주는 방식

콘텐츠 특징 선정

기존 데이터에서 특징 선정

가게			store
가게식별자	id	BIGINT	NOT NULL
주소	address	VCHAR(255)	NOT NULL
동네	area	VCHAR(255)	NOT NULL
지점	branch	VCHAR(255)	NULL
업종	category	VCHAR(255)	NULL
가게사진	image	VCHAR(255)	NULL
위도	latitude	BIGINT	NOT NULL
경도	longitude	BIGINT	NOT NULL
가게명	store_name	VCHAR(255)	NOT NULL
전화번호	tel	VCHAR(255)	NULL

리뷰			Review
리뷰식별자	id	BIGINT	NOT NULL
점수	score	INT	NOT NULL
리뷰사진	review_image	VCHAR(255)	NULL
내용	content	TEXT	NOT NULL
작성날짜	reg_time	DATETIME	NOT NULL
가게식별자	id2	BIGINT	NOT NULL
사용자식별자	id3	BIGINT	NOT NULL

사용자			User
사용자식별자	id	BIGINT	NOT NULL
나이	born_year	INT	NOT NULL
이메일	email	VCHAR(255)	NOT NULL
성별	gender	VCHAR(255)	NOT NULL
닉네임	nickname	VCHAR(255)	NOT NULL

콘텐츠 특징 선정

알고리즘을 이용한 콘텐츠 특징 추출

=> 사용자 리뷰를 기반 으로 가게의 특징을 뽑아낸다.

리뷰 예시) 진하고 얼큰한 국물에 햄과 소시지가 푸짐하고 맛있는 부대찌개 집

특징이 될 만한 키워드 추출 + 수치화

TF-IDF

Term Frequency

단어 빈도수

×

Inverse Document Frequency

역문서 빈도

전체 문서들에서 특정 단어가 얼마나 나오는가?

TF-IDF

예시

문서	내용
0	먹고 싶은 사과
1	먹고 싶은 바나나
2	길고 노란 바나나 바나나
3	저는 과일이 좋아요

자연어 처리를 통해
키워드로 나눔

TF(Term Frequecy)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	1	0	1	1	0	0
문서2	0	0	0	1	1	0	1	0	0
문서3	0	1	1	0	2	0	0	0	0
문서4	1	0	0	0	0	0	0	1	1

DF(Document Frequecy)

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
총합	1	1	1	2	3	1	2	1	1

TF-IDF

유사도 계산 방식을 사용

	과일이	길고	노란	먹고	바나나	사과	싶은	저는	좋아요
문서1	0	0	0	0.2876	0	0.6931	0.2876	0	0
문서2	0	0	0	0.2876	0	0	0.2876	0	0
문서3	0	0.6931	0.6931	0	0	0	0	0	0
문서4	0.6931	0	0	0	0	0	0	0.6931	0.6931

IDF

단어	IDF(역 문서 빈도)
과일이	ln(4/(1+1)) = 0.693147
길고	ln(4/(1+1)) = 0.693147
노란	ln(4/(1+1)) = 0.693147
먹고	ln(4/(2+1)) = 0.287682
바나나	ln(4/(3+1)) = 0
사과	ln(4/(1+1)) = 0.693147
싶은	ln(4/(2+1)) = 0.287682
저는	ln(4/(1+1)) = 0.693147
좋아요	ln(4/(1+1)) = 0.693147

$$idf(d,t) = log(\frac{n}{1+df(t)})$$

idf값이 너무 커지는 것을 막기 위해
log 값을 사용해준다.

04 협업 필터링

- 메모리 기반 방식
- 모델 기반 방식

협업 필터링(Collaborative Filtering)

: 대규모의 기존 사용자로부터 모은 데이터(평점, 구매 패턴 등)를 기반으로 사용자와 비슷한 성향의 항목을 추천하는 기술

Memory-based Methods

사용자 기반 협업 필터링(User-based Collaborative filtering)

사용자 간의 선호도를 분석하여 나와 유사한 성향의 사용자가 좋아한 상품/콘텐츠를 추천하는 기법

아이템 기반 협업 필터링(Item-based Collaborative Filtering)

사용자들의 선호도를 바탕으로 아이템 간의 유사도를 계산하고, 특정 사용자가 어떤 아이템을 구매하거나 좋다고 평가하면 그와 유사한 아이템을 추천해주는 방식 -> 대상이 사람이 아닌 아이템

Model-based Methods

평점을 예측할 수 있는 모델을 만드는 방식

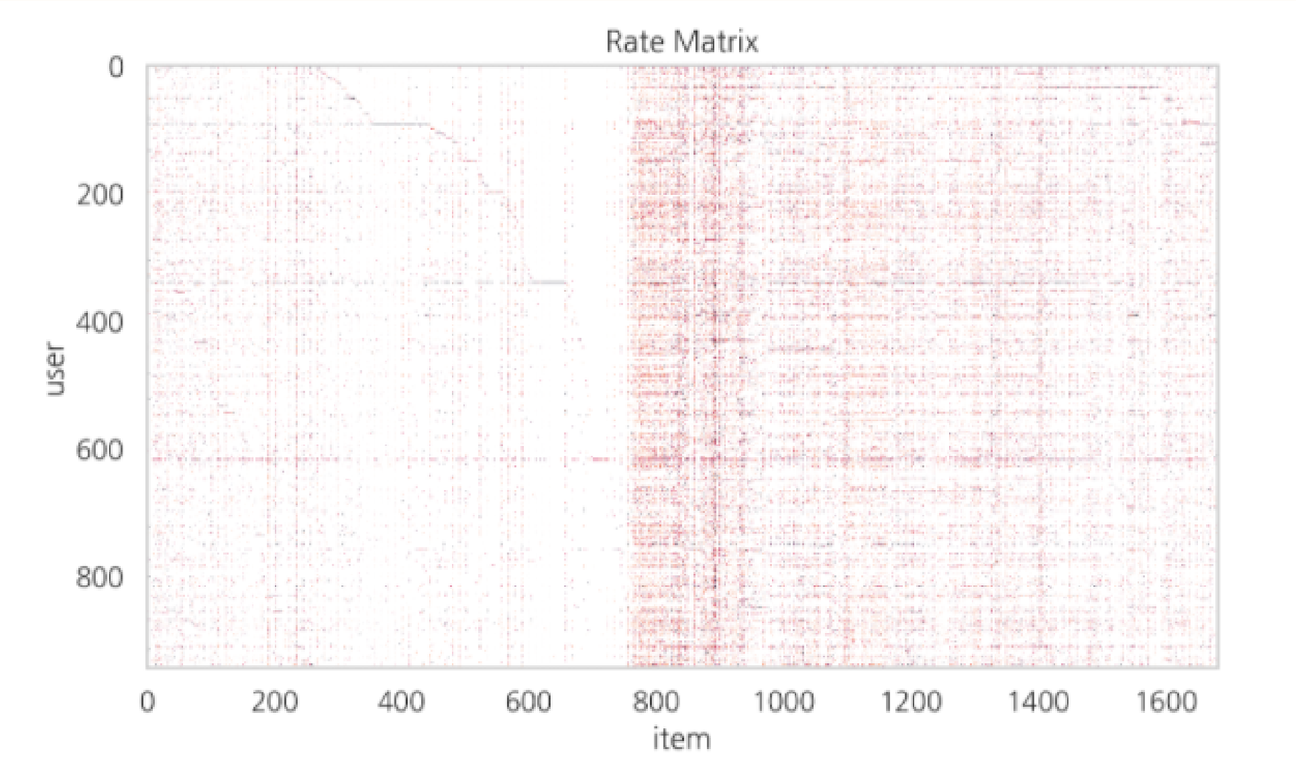
일단 협업 필터링을 하기 위해서는 User-Item 행렬이 필요

User-item 행렬

User - Store 평점 행렬 생성

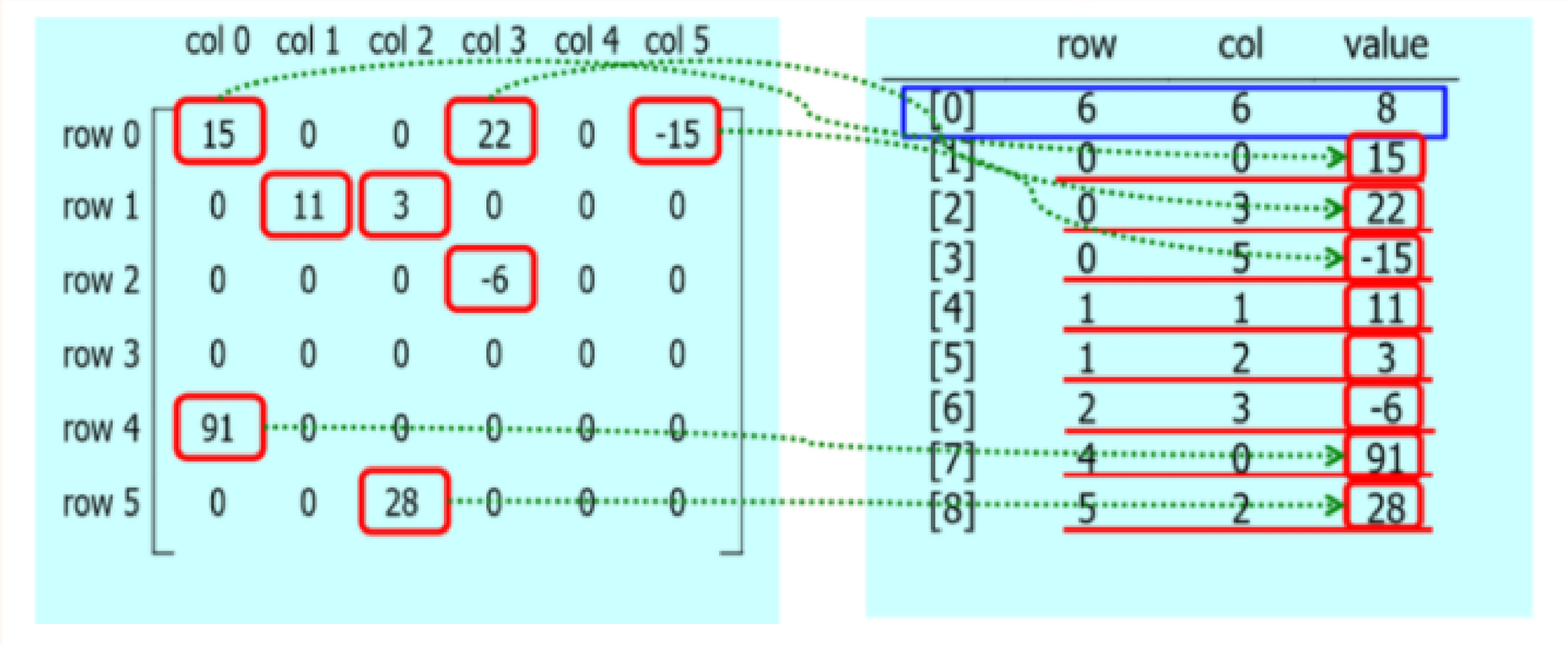
리뷰			Review
리뷰식별자	id	BIGINT	NOT NULL
점수	score	INT	NOT NULL
리뷰사진	review_image	VCHAR(255)	NULL
내용	content	TEXT	NOT NULL
작성날짜	reg_time	DATETIME	NOT NULL
가게식별자	id2	BIGINT	NOT NULL
사용자식별자	id3	BIGINT	NOT NULL

user(약1.9만명) * store(약46만개)
=> 약 87억4천만 크기의 평점 matrix 생성
But) 리뷰 개수는 91398개



희소 행렬(sparse matrix)

희소 행렬



Python의 경우 Scipy라이브러리에서 희소행렬을 효율적으로
저장하기 위한 방법들(COO, CSR, DOK)을 제공

메모리 기반 필터링(Neighborhood Model)

유사한 것을 추천해준다.

	item1	item2	item3	item4	item5	itme6	평균	cosine (i,3)	pearson (i,3)
user1	7	6	7	4	5	4	5.5	0.956	0.894
user2	6	7		4	3	4	4.8	0.981	0.939
user3		3	3	1	1		2	1.0	1.0
user4	1	2	2	3	3	4	2.5	0.789	-1.0
user5	1		1	2	3	3	2	0.645	-0.817

=> KNN 알고리즘을 사용하여 유사한 몇 명을 선택할 지 결정

메모리 기반 필터링(Neighborhood Model)

유사한 것을 추천해준다.

	item1	itme6	평균	pearson (i,3)
user1	7	4	5.5	0.894
user2	6	4	4.8	0.939
user3			2	1.0
user4	1	4	2.5	-1.0
user5	1	3	2	-0.817

유사한 사용자에 대한 값으로만 계산

$$\hat{r}_{31} = \frac{7 * 0.894 + 6 * 0.939}{0.894 + 0.939} \approx 6.49$$

$$\hat{r}_{36} = \frac{4 * 0.894 + 4 * 0.939}{0.894 + 0.939} = 4$$

사용자의 평균치를 포함하여 유사한 사용자의 값을 통해 계산

사용자1의 아이템1의 평점 - 사용자1의 평균 평점

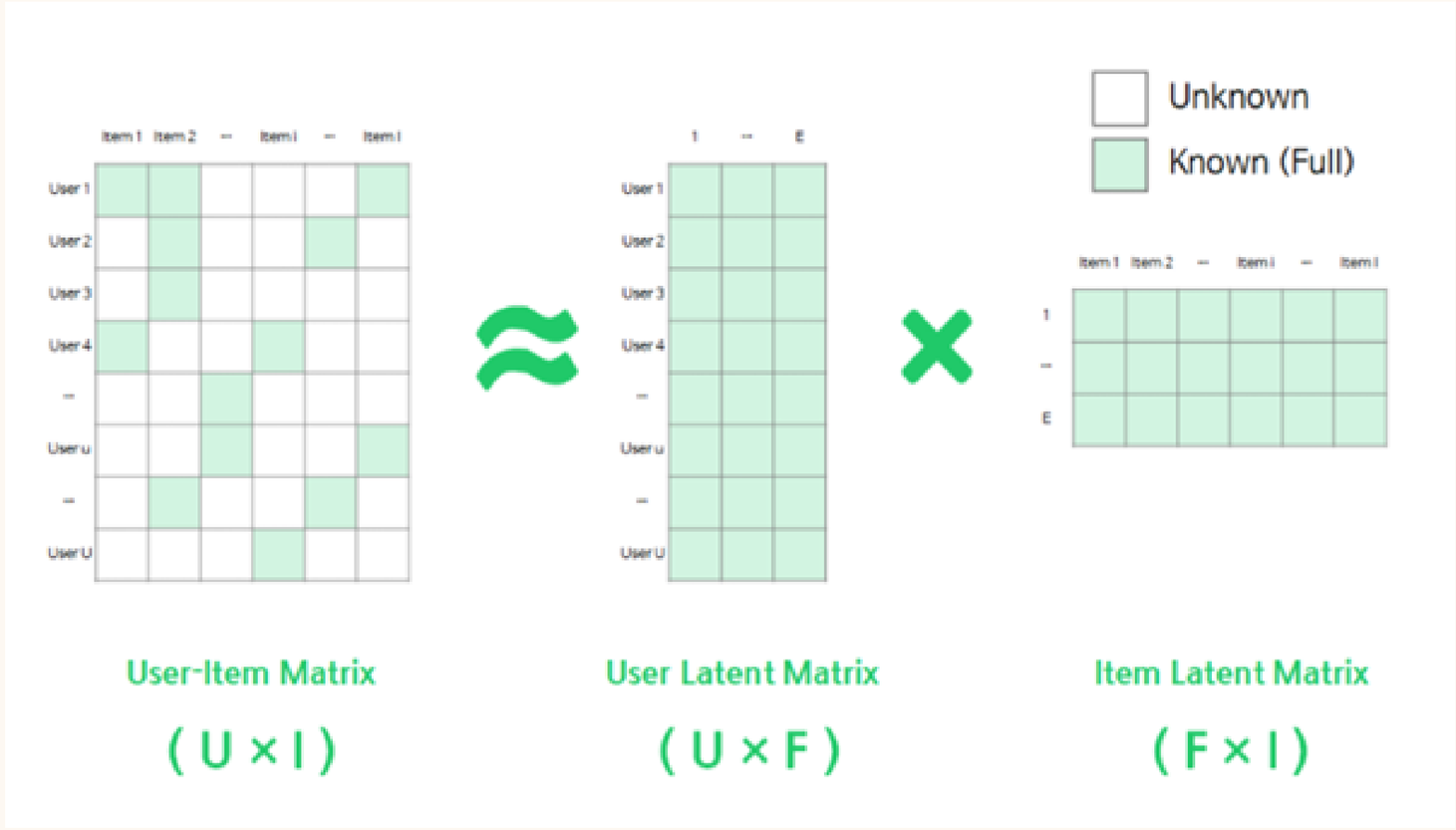
$$\hat{r}_{31} = 2 + \frac{1.5 * 0.894 + 1.2 * 0.939}{0.894 + 0.939} \approx 3.35$$

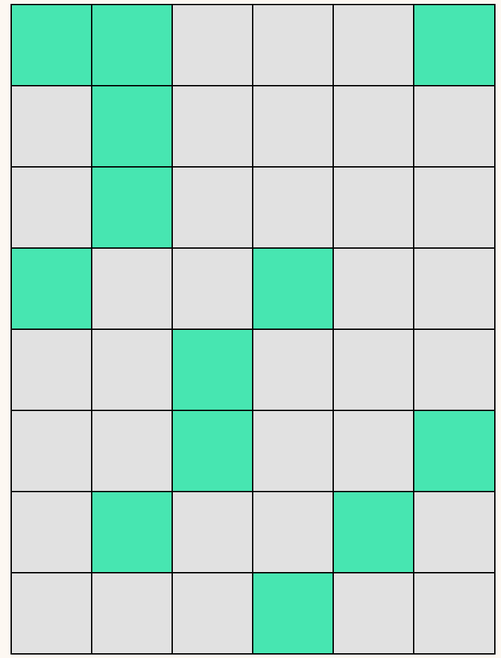
$$\hat{r}_{36} = 2 + \frac{-1.5 * 0.894 - 0.8 * 0.939}{0.894 + 0.939} \approx 0.86$$

사용자3의 평균 평점

모델 기반 필터링(Latent Factor)

행렬 분해(Matrix Fatorization)





User-Item 예측 Matrix
(U x I)

최적화(Optimization) 문제

: 실제 값과 예측 값의 오차를 최소화 나가는 방식

SGD(Stochastic Gradient Descent)

:기본적인 방식은 GD와 같으나 오차를 계산할 때 전체 데이터중 일부만 가지고 계산하여 일반적인 GD 방식 보다 빠름
User latent와 Item Latent를 동시에 최적화

ALS(Alternating Least Square)

:오차를 줄이기 위해 User latent와 Item Latent중 하나의 값을 고정시키고 나머지 하나만 최적화, 이 과정을 번갈아 가면서 반복하면서 최적화

감사합니다

Q&A