추천시스템

- 권수태 교수

JEONJU UNIVERSITY
The Place for Superstars

1. 추천시스템

❖추천시스템 개요

- 현재 다양한 곳에서 추천 시스템을 통해 고객을 자신의 사이트에 오래 머무르게 하기 위해 전력을 기울이고 있고, 추천 시스템은 온라인에서 그 진가를 더 잘 발휘하고 있음(전자상거래 업체, 콘텐츠 업체)
- 사용자가 무엇을 원하는지 빠르게 찾아내 사용자의 온라인 쇼핑의 즐거움을 느끼게 하는 것이 필요

너무 많은 상품으로 가득찬 온라인 스토어

VS

추천엔진은 사용자가 무엇을 원하는지 빠르게 찾아내어 사용자의 온라인 쇼핑 이용 즐거움을 배가시킨다.

한정된 시간, 어떤 상품을 골라야 할지 선택의 압박

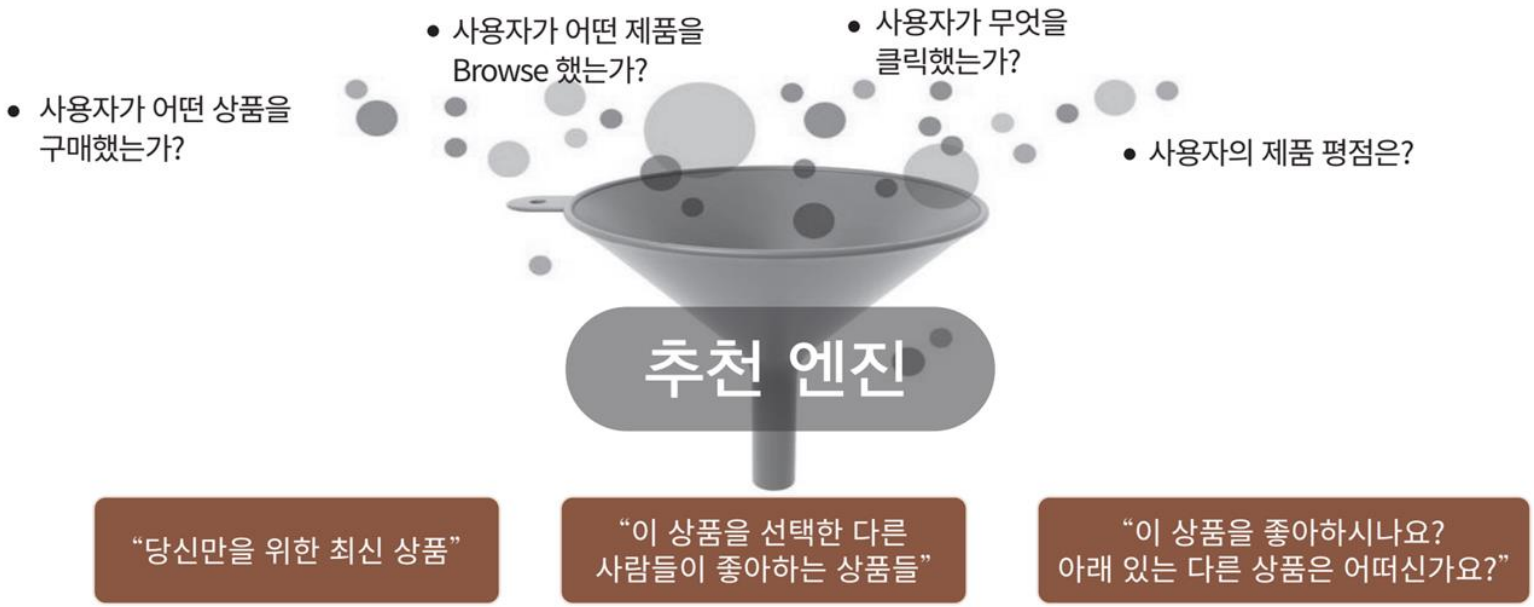
〈 추천 시스템으로 사용자의 선택 부담을 해결 〉

JEONJU UNIVERSITY

1. 추천시스템

❖추천시스템 개요

- ➢ 온라인 스토어는 많은 양의 고객과 상품 관련 데이터를 가짐
- ➢ 이 데이터를 이용해 상품을 추천하는데 사용



〈 데이터 기반의 추천 시스템 〉

1. 추천시스템

❖추천시스템 개요

- 하나의 콘텐츠를 선택했을 때 선택된 콘텐츠와 연계된 추천 콘텐츠가 얼마나 사용자의 관심을 끌고 개인에게 맞춘 콘텐츠를 추천했는지는 그 사이트의 평판을 좌우하는 매우 중요한 요소
  - ✓ 추천시스템의 진정한 묘미는 사용자 자신도 좋아하는지 몰랐던 취향을 시스템이 발견하고 그에 맞는 콘텐츠를 추천해주는 것
  - ✓ 이러한 추천 시스템을 접한 사용자는 해당 사이트를 더 강하게 신뢰하게 되어 더 많은 추천 콘텐트를 선택하게 됨

YouTube KR
시청 영상의 70%
추천 동영상

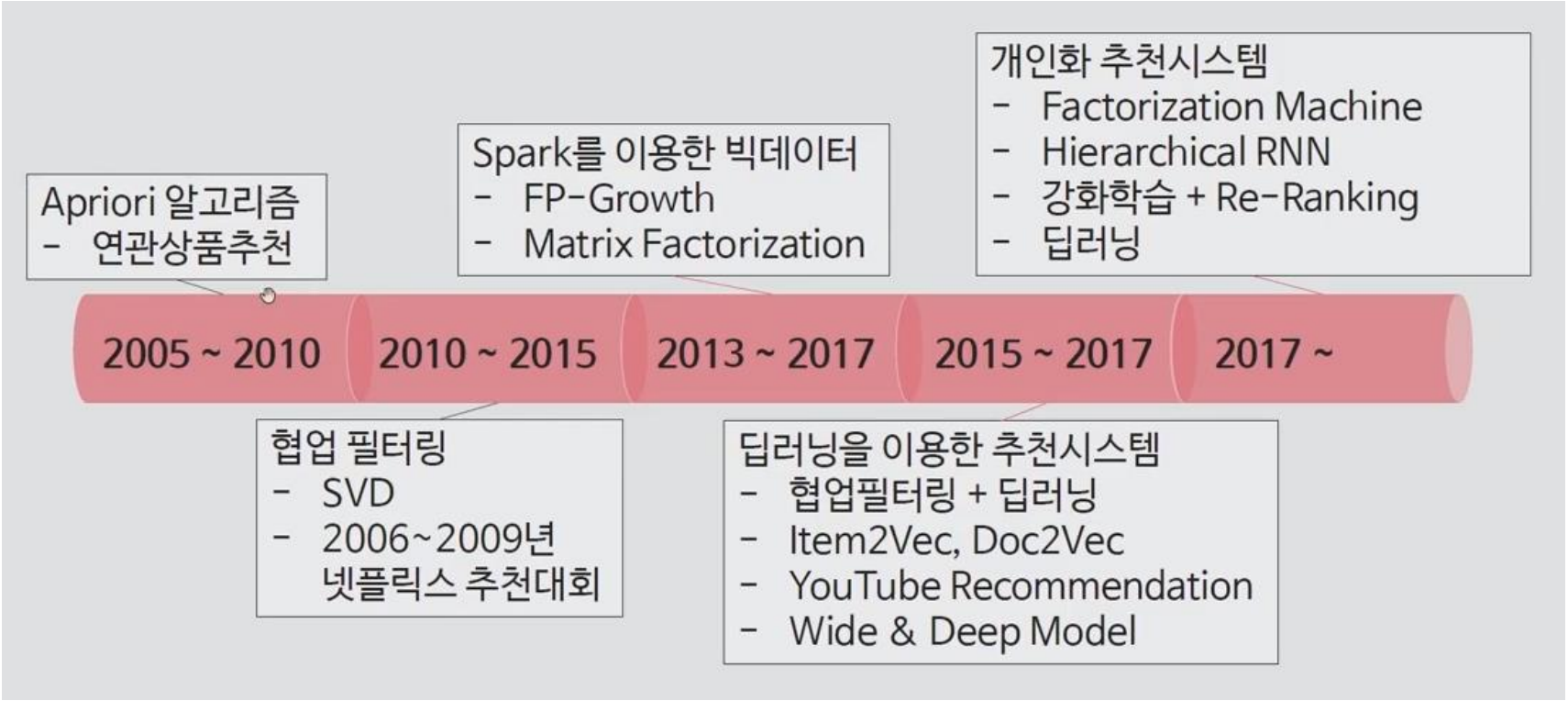
NETFLIX
시청 영상의 80%
추천 동영상

Spotify
서비스 전체(100%)
추천 음악

넷플릭스는 90초 골든타임이라는 말을 사용

1. 추천시스템

❖추천시스템 개요



1. 추천시스템

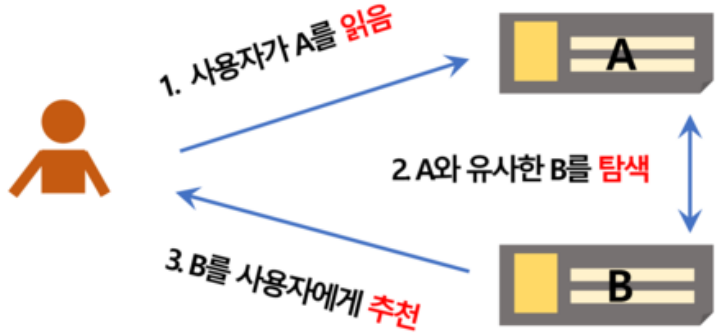
❖추천시스템 유형

- 추천 시스템은 콘텐츠 기반 필터링 방식과 협업 필터링 방식(최근접 이웃, 잠재 요인) 으로 나뉨
- 초창기에는 콘텐츠 기반, 최근접 이웃 기반을 주로 사용했지만 현재는 잠재 요인 협업 필터링을 적용
- 하지만 아이템의 특성에 따라 콘텐츠, 최근접을 유지하는 경우도 있으며, 하이브리드 형식으로 콘텐츠+협업 결합을 통해 사용하는 경우도 존재

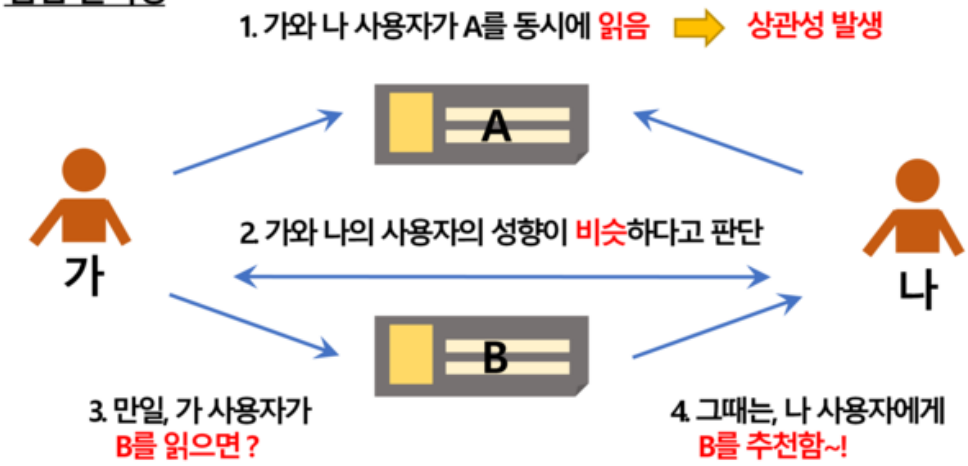
1. 추천시스템

❖추천시스템 유형

콘텐츠 기반 필터링



협업 필터링



[참조: 이재호]

JEONJU UNIVERSITY

1. 추천시스템

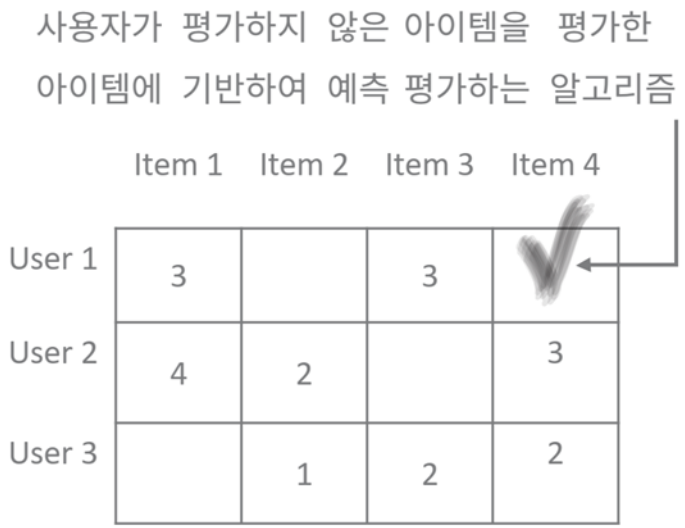
❖콘텐츠 기반 필터링 추천시스템

- 사용자가 특정한 아이템을 매우 선호하는 경우, 그 아이템과 비슷한 콘텐츠를 가진 다른 아이템을 추천하는 방식
  - ✓아이템을 구성하는 다양한 콘텐츠들을 텍스트 기반 문서 유사도로 비교하여 추천
  - ✓ex) 영화를 추천하는 경우를 예시로 들면 사용자가 선호하는 감독, 장르, 키워드 등의 콘텐츠를 다양하게 포함하고 있으므로 콘텐츠 기반 필터링 추천 시스템에 의해 특정 영화가 추천될 수 있음
  - ✓과정 : 영화 구성 콘텐츠 텍스트 -> 피처 벡터화 -> 코사인 유사도 -> 유사도, 평점에 따른 영화 추천

1. 추천시스템

❖최근접 이웃 협업 필터링

- 협업 필터링: 사용자가 아이템에 매긴 평점 정보나 상품 구매 이력과 같은 사용자 행동 양식만을 기반으로 추천을 수행하는 것
- 주요 목표는 사용자-아이템 평점 매트릭스와 같은 축적된 사용자 행동 데이터를 기반으로 사용자가 아직 평가하지 않은 아이템을 예측 평가하는 것



JEONJU UNIVERSITY

1. 추천시스템

❖최근접 이웃 협업 필터링

- ➢ 사용자(행)-아이템(열) 평점 행렬 데이터에 의지해 추천을 수행함
- ➢ 다른 형태의 데이터로 되어있는 경우 pivot() 함수 등을 이용해 행렬 평점 데이터로 변환해야함

로우 레벨 형태의 사용자 - 아이템 평점 데이터

User ID	Item ID	Rating
User 1	Item 1	3
User 1	Item 3	3
User 2	Item 1	4
User 2	Item 2	1
User 3	Item 4	5

변환

사용자 로우, 아이템 칼럼으로 구성된
사용자 - 아이템 평점 데이터

	Item 1	Item 2	Item 3	Item 4
User 1	3		3	
User 2	4	1		
User 3				5

1. 추천시스템

❖최근접 이웃 협업 필터링

- 사용자-아이템 평점 행렬은 많은 아이템을 열로 가지는 다차원 행렬이며, 사용자가 모든 아이템에 대해 평점을 매기는 경우가 거의 없기 때문에 희소 행렬 특성을 가짐
- 최근접 이웃은 사용자 기반과 아이템 기반으로 다시 나뉨

1. 추천시스템

❖최근접 이웃 협업 필터링

➢ 사용자 기반 :

✓ 비슷한 고객들은 이 상품을 구매

✓ 특정 사용자와 유사한 다른 사용자를 TOP-N으로 선정해 이 TOP-N 사용자가 좋아하는 아이템을 추천하는 방식

✓ 즉, 특정사용자와 타 사용자간의 유사도를 측정한 뒤 가장 유사도가 높은 TOP-N 사용자를 추출해 그들이 선호하는 아이템을 추천하는 것

		다크 나이트	인터스텔라	엣지오브 투모로우	프로메테우스	스타워즈 라스트제다이
상호간 유사도 높음	사용자 A	5	4	4		
	사용자 B	5	3	4	5	3
	사용자 C	4	3	3	2	5

사용자 A는 사용자 C 보다 사용자 B와 영화 평점 측면에서 유사도가 높음. 따라서 사용자 A 에게는 사용자 B가 재미있게 본 '프로메테우스' 를 추천

1. 추천시스템

❖최근접 이웃 협업 필터링

➢ 아이템 기반 :

- ✓ 이 상품을 선택한 다른 고객은 다음 상품도 구매
- ✓ 아이템이 가지는 속성과는 상관없이 사용자들이 그 아이템을 좋아하는지/싫어하는 지의 평가 척도가 유사한 아이템을 추천하는 기준이 되는 알고리즘
- ✓ 행이 아이템이고 열이 사용자인 행렬 데이터 사용
- ✓ 사용자 기반보다 정확도가 더 높은 편

		사용자 A	사용자 B	사용자 C	사용자 D	사용자 E
상호간 유사도 높음	다크 나이트	5	4	5	5	5
	프로메테우스	5	4	4		5
	스타워즈 라스트제다이	4	3	3		4

여러 사용자들의 평점을 기준으로 볼 때 '다크 나이트' 와 가장 유사한 영화는 '프로메테우스'

1. 추천시스템

❖잠재요인 협업 필터링

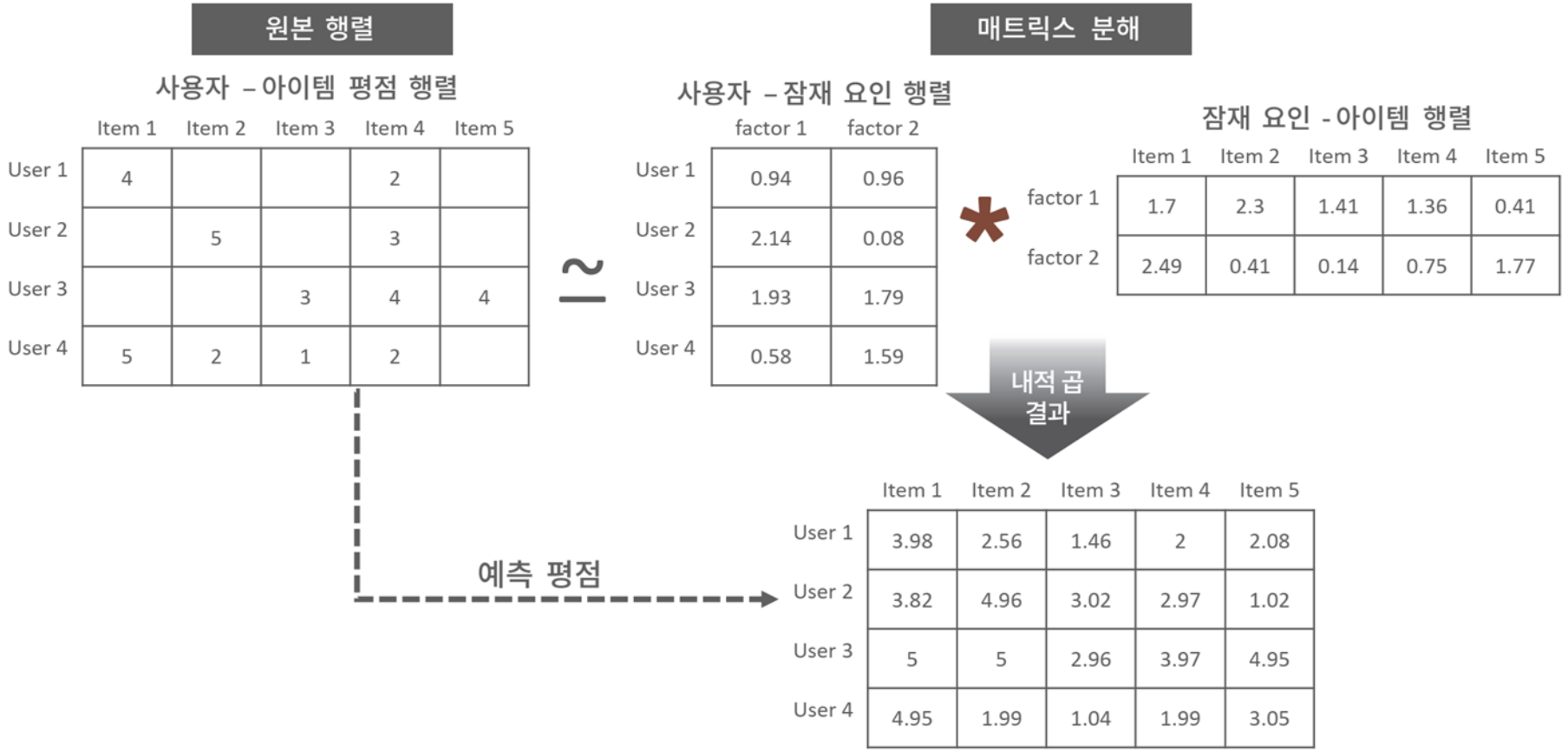
- 사용자-아이템 평점 행렬속에 숨어있는 잠재요인을 추출해 추천 예측을 할 수 있게 하는 기법
- 행렬 분해: 차원 감소 기법으로 분해하는 과정에서 잠재 요인 추출

- 잠재 요인을 기반으로 다차원 희소 행렬을 저차원 밀집 행렬의 사용자-잠재 요인 행렬과 아이템-잠재 요인 행렬의 전치 행렬로 분해

-> 이렇게 분해된 두 행렬의 내적을 통해 새로운 예측 사용자-아이템 평점 행렬 데이터를 만들어 사용자가 아직 평점을 부여하지 않은 아이템에 대한 예측 평점을 생성하는 것이 원리

1. 추천시스템

❖잠재요인 협업 필터링



〈 행렬 분해를 통한 잠재 요인 협업 필터링 〉

JEONJU UNIVERSITY

1. 추천시스템

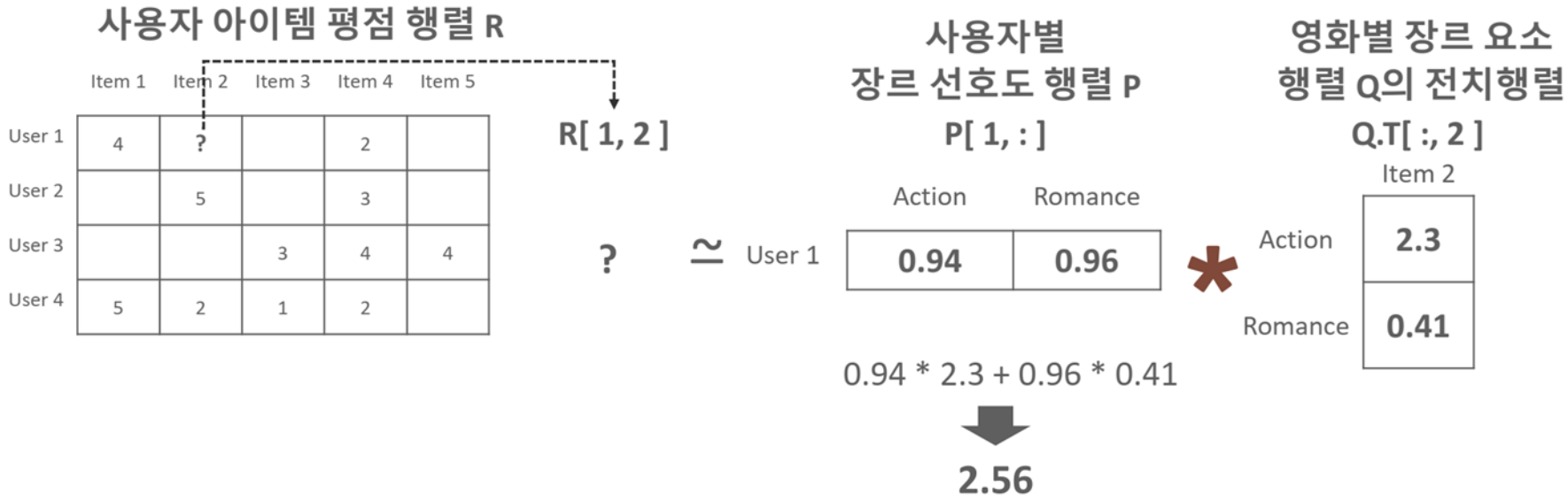
❖잠재요인 협업 필터링

- ➢ 사용자-잠재 요인 행렬 : 사용자의 선호도
- ➢ 아이템-잠재 요인 행렬 : 아이템의 특성값

1. 추천시스템

❖잠재요인 협업 필터링

- ➢ 사용자-잠재 요인 행렬 : 사용자의 선호도
- ➢ 아이템-잠재 요인 행렬 : 아이템의 특성값



1. 추천시스템

❖잠재요인 협업 필터링

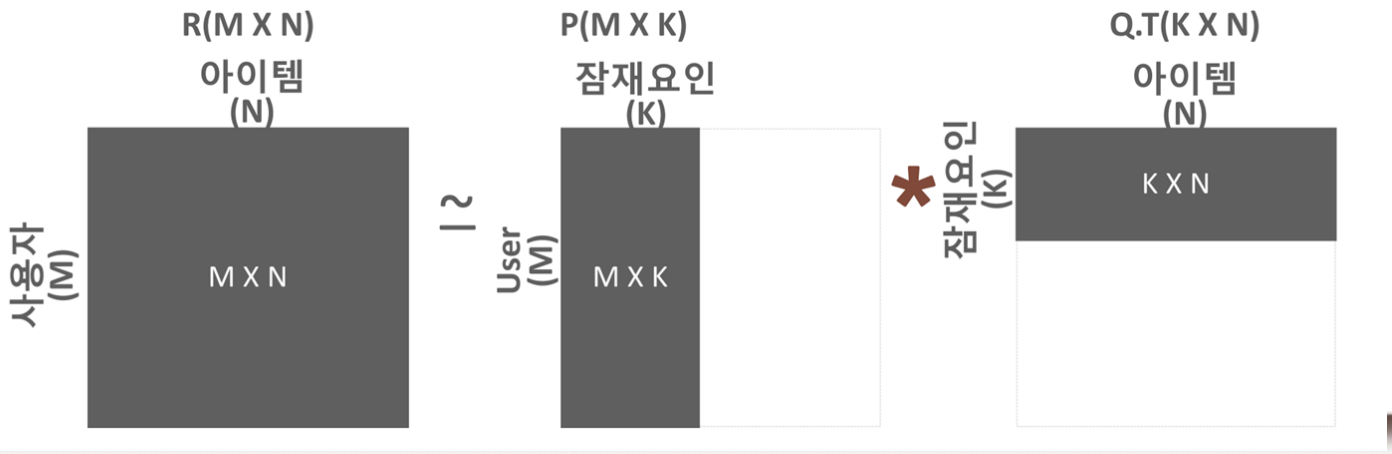
➢ 행렬 분해

✓ 다차원의 매트릭스를 저차원의 매트릭스로 분해하는 기법

✓ 쉽게 말하면 인수분해 하는 것처럼 행렬을 대상으로 분해하는 것

✓ SVD(Singular Vector Decomposition), NMF(Non-Negative Matrix Factorization) 등이 존재

✓ 사용자-아이템 행렬 R은 M x N 차원으로 구성, 이는 행렬 분해를 통해 사용자-잠재 요인 행렬P (M X k 차원) 과 잠재 요인-아이템 행렬 Q.T (K x N 차원)으로 분해 가능



JEONJU UNIVERSITY

1. 추천시스템

❖잠재요인 협업 필터링

➢ 행렬 분해

R(4 X 5)

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4			2	
User 2		5		3	
User 3			3	4	4
User 4	5	2	1	2	

≈

P(4 X 2)

	factor 1	factor 2
User 1	0.94	0.96
User 2	2.14	0.08
User 3	1.93	1.79
User 4	0.58	1.59

개별 Row는 개별 User에 대한 Latent Factor들을 반영

*

Q.T(2 X 4)

	Item 1	Item 2	Item 3	Item 4	Item 5
factor 1	1.7	2.3	1.41	1.36	0.41
factor 2	2.49	0.41	0.14	0.75	1.77

개별 Column은 개별 item에 대한 Latent Factor 들을 반영

1. 추천시스템

❖잠재요인 협업 필터링

➢행렬 분해

$$r_{(u,i)} = p_u * q_i^t$$

r(2, 4)

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4				
User 2				**3**	
User 3			3	4	4
User 4	5	2	1	2	

$\simeq$

p_2

	factor 1	factor 2
User 1	0.94	0.96
User 2	**2.14**	**0.08**
User 3	1.93	1.79
User 4	0.58	1.59

$*$

q_4^t

	Item 1	Item 2	Item 3	Item 4	Item 5
factor 1	1.7	2.3	1.41	**1.36**	0.41
factor 2	2.49	0.41	0.14	**0.75**	1.77

$\hat{r}$ = 2.14 * 1.36 + 0.08 * 0.75 = 2.97

r(2, 3)

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	4			2	
User 2			**?**	3	
User 3			3	4	4
User 4	5	2	1	2	

$\simeq$

p_2

	factor 1	factor 2
User 1	0.94	0.96
User 2	**2.14**	**0.08**
User 3	1.93	1.79
User 4	0.58	1.59

$*$

q_3^t

	Item 1	Item 2	Item 3	Item 4	Item 5
factor 1	1.7	2.3	**1.41**	1.36	0.41
factor 2	2.49	0.41	**0.14**	0.75	1.77

$\hat{r}_{(2,3)}$ = 2.14 * 1.41 + 0.08 * 0.14 = 3.02

JEONJU UNIVERSITY

1. 추천시스템

❖잠재요인 협업 필터링

➢행렬 분해

$$R \simeq \hat{R} = P * Q.T$$

P

	factor 1	factor 2
User 1	0.94	0.96
User 2	2.14	0.08
User 3	1.93	1.79
User 4	0.58	1.59

*

Q.T

	Item 1	Item 2	Item 3	Item 4	Item 5
factor 1	1.7	2.3	1.41	1.36	0.41
factor 2	2.49	0.41	0.14	0.75	1.77

예측 평점

예측 유저-아이템 Rating 매트릭스 $\hat{R}$

	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	3.98	2.56	1.46	2	2.08
User 2	3.82	4.96	3.02	2.97	1.02
User 3	5	5	2.96	3.97	4.95
User 4	4.95	1.99	1.04	1.99	3.05

1. 추천시스템

❖잠재요인 협업 필터링

- ➢그렇다면 R 행렬을 어떻게 P 와 Q 로 분해할까?
  - ✓행렬 분해는 보통 SVD(Singular Value Decomposition) 방식을 이용하지만 이는 null값이 없는 행렬에만 적용이 가능
  - ✓하지만 사용자-아이템 행렬에는 null 값이 많기 때문에 불가능
  - ✓확률적 경사 하강법이나 ALS(Alternating Least Square) 방식을 이용해 SVD 를 수행

1. 추천시스템

❖잠재요인 협업 필터링

➢확률적 경사 하강법을 이용한 행렬분해

✓ 회귀에서 언급된 경사 하강법의 한 종류

✓ P와 Q 행렬로 계산된 예측 R 행렬 값이 실제 R 행렬 값과 가장 최소의 오류를 가질 수 있도록 반복적인 비용 함수 최적화를 통해 P와 Q를 유추해내는 것

✓ 과정

1) P와 Q 를 임의의 값을 가진 행렬로 설정

2) P와 Q.T값을 곱해 예측 R 행렬을 계산하고 예측 R 행렬과 실제 R 행렬에 해당하는 오류 값을 계산함

3) 이 오류 값을 최소화할 수 있도록 P와 Q 행렬을 적절한 값으로 각각 업데이트

4) 만족할 만한 오류 값을 가질 때까지 2,3번 작업을 반복하며 P와 Q 값을 업데이트해 근사화 함

1. 추천시스템

❖잠재요인 협업 필터링

➢ 확률적 경사 하강법을 이용한 행렬분해

✓ 실제값과 예측값의 오류 최소화와 L2 규제를 고려한 비용 함수식

$$min\sum(r_{(u,i)} - p_u q_i^t)^2 + \lambda(\| q_i \|^2 + \| p_u \|^2)$$

$$p'_u = p_u + \eta(e_{(u,i)} * q_i - \lambda * p_u)$$
$$q'_i = q_i + \eta(e_{(u,i)} * p_u - \lambda * q_i)$$

✓ SGD 학습률, 실제 행렬값과 예측 행렬 값의 차이오류, L2 규제 계수

- L2 규제를 반영해 실제 R 행렬값과 예측 R 행렬 값의 차이를 최소화하는 방향성을 가지고 P 행렬과 Q 행렬에 업데이트 값을 반복적으로 수행하면서 최적화된 예측 R 행렬을 구하는 방식

1. 추천시스템

❖잠재요인 협업 필터링

➢ 확률적 경사 하강법을 이용한 행렬분해

```
import numpy as np

# 원본 행렬 R 생성, 분해 행렬 P와 Q 초기화, 잠재요인 차원 K는 3 설정.
R = np.array([[4, np.NaN, np.NaN, 2, np.NaN ],
              [np.NaN, 5, np.NaN, 3, 1 ],
              [np.NaN, np.NaN, 3, 4, 4 ],
              [5, 2, 1, 2, np.NaN ]])
num_users, num_items = R.shape
K=3

# P와 Q 매트릭스의 크기를 지정하고 정규분포를 가진 random한 값으로
입력합니다.
np.random.seed(1)
P = np.random.normal(scale=1./K, size=(num_users, K))
Q = np.random.normal(scale=1./K, size=(num_items, K))
```


1. 추천시스템

❖잠재요인 협업 필터링

➢ 확률적 경사 하강법을 이용한 행렬분해

```
from sklearn.metrics import mean_squared_error

def get_rmse(R, P, Q, non_zeros):
    error = 0
    # 두개의 분해된 행렬 P와 Q.T의 내적으로 예측 R 행렬 생성
    full_pred_matrix = np.dot(P, Q.T)

    # 실제 R 행렬에서 널이 아닌 값의 위치 인덱스 추출하여 실제 R 행렬과 예측 행렬의 RMSE 추출
    x_non_zero_ind = [non_zero[0] for non_zero in non_zeros]
    y_non_zero_ind = [non_zero[1] for non_zero in non_zeros]
    R_non_zeros = R[x_non_zero_ind, y_non_zero_ind]
    full_pred_matrix_non_zeros = full_pred_matrix[x_non_zero_ind, y_non_zero_ind]

    mse = mean_squared_error(R_non_zeros, full_pred_matrix_non_zeros)
    rmse = np.sqrt(mse)

    return rmse
```


1. 추천시스템

❖잠재요인 협업 필터링

➢ 확률적 경사 하강법을 이용한 행렬분해

```
# R > 0 인 행 위치, 열 위치, 값을 non_zeros 리스트에 저장.
non_zeros = [ (i, j, R[i,j]) for i in range(num_users) for j in range(num_items) if R[i,j] > 0 ]

steps=1000
learning_rate=0.01
r_lambda=0.01

# SGD 기법으로 P와 Q 매트릭스를 계속 업데이트.
for step in range(steps):
    for i, j, r in non_zeros:
        # 실제 값과 예측 값의 차이인 오류 값 구함
        eij = r - np.dot(P[i, :], Q[j, :].T)
        # Regularization을 반영한 SGD 업데이트 공식 적용
        P[i,:] = P[i,:] + learning_rate*(eij * Q[j, :] - r_lambda*P[i,:])
        Q[j,:] = Q[j,:] + learning_rate*(eij * P[i, :] - r_lambda*Q[j,:])

    rmse = get_rmse(R, P, Q, non_zeros)
    if (step % 50) == 0 :
        print("### iteration step : ", step," rmse : ", rmse)
```


1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

- https://www.kaggle.com/tmdb/tmdb-movie-metadata 캐글의 TMDB 5000 영화 데이터 세트를 이용해 실습(2개 파일 다운로드)

- 장르속성을 이용한 영화콘텐츠 기반 필터링
  - ✓ 콘텐츠 기반 필터링은 사용자가 특정 영화를 감상하고 그 영화를 좋아했다면 그 영화와 비슷한 특성, 속성, 구성요소를 가진 다른 영화를 추천하는 것
  - ✓ 영화 간의 유사성 판단을 하는 기준이 영화를 구성하는 콘텐츠(장르, 감독, 배우, 평점) 등등이 됨
  - ✓ 이번 실습에서는 다양한 콘텐츠들 중 영화 장르 속성을 기반으로, 장르 컬럼값의 유사도를 비교한 뒤 그중 높은 평점을 가지는 영화를 추천하는 방식

1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

➢ 데이터 로딩 및 가공

✓ 파이썬 리스트 내부에 여러 개의 딕셔너리가 있는 형태의 문자열

- 컬럼의 문자열을 분해해서 개별 장르를 파이썬 리스트 객체로 추출
- 파이썬 ast 모듈의 literal_eval() 함수 이용하여 list[dict1, dict2] 객체로 만듬
- Series 객체의 apply() 에 literal_eval() 함수 적용

```
from ast import literal_eval
movies_df['genres'] = movies_df['genres'].apply(literal_eval)
movies_df['keywords'] = movies_df['keywords'].apply(literal_eval)
```

- 리스트내 여러 개의 딕셔너리의 name 키에 해당하는 값을 찾아 이를 리스트 객체로 변환

```
movies_df['genres'] = movies_df['genres'].apply(lambda x : [ y['name'] for y in x])
movies_df['keywords'] = movies_df['keywords'].apply(lambda x : [ y['name'] for y in x])
movies_df[['genres', 'keywords']][:1]
```


1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

- 장르 콘텐츠 유사도 측정
  - ✓ 텍스트 문자를 1차 가공했으니 이제 장르를 이용해 코사인 유사도를 비교
  - ✓ 과정
    - 1) 장르 칼럼을 Count 기반으로 피처 벡터화 변환
    - 2) 변환한 데이터 세트를 코사인 유사도를 통해 비교함
    - 3) 장르 유사도가 높은 영화 중 평점이 높은 순으로 영화를 추천함
  - ✓ 리스트 객체내의 개별 값을 연속된 문자열로 변환하려면 일반적으로 ('구분문자').join(리스트 객체)를 사용하면 됨

  => 개별 요소를 공백 문자로 구분하는 문자열로 변환해 별도의 칼럼을 생성

  - ✓ 이렇게 생성된 피처 벡터 행렬을 이용해 코사인 유사도를 계산
  - ✓ 장르 기준으로 콘텐츠 기반 필터링을 수행하기 위해 위의 genre_sim 객체를 이용
  - ✓ 기준 행렬로 비교 대상이 되는 행의 유사도 값이 높은 순으로 정렬된 행렬의 위치 인덱스 값을 추출

1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

➢ 장르 콘텐츠 유사도 측정

기준행			
0	1	7	2
1	1	2	4
2	0	8	3
3	2	0	3

cosine_similarities(적용)

비교 대상 행

기준행	0	1	2	3
0	1 (0 행 자신의 유사도)	0.68 (0행과 1행의 유사도)	0.99 (0행과 2행의 유사도)	0.3 (0행과 3행의 유사도)
1	0.68 (1행과 0행의 유사도)	1 (1행 자신의 유사도)	0.72 (1행과 2행의 유사도)	0.85 (1행과 3행의 유사도)
2	0.99 (2행과 0행의 유사도)	0.72 (2행과 1행의 유사도)	1 (2행 자신의 유사도)	0.29 (2행과 3행의 유사도)
3	0.3 (3행과 0행의 유사도)	0.85 (3행과 1행의 유사도)	0.29 (3행과 2행의 유사도)	1 (3행 자신의 유사도)

1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

- 장르 콘텐츠 필터링을 이용한 영화 추천
  - ✓ 장르 유사도에 따라 영화를 추천하는 함수를 생성
  - ✓ The Godfather 이라는 영화와 유사한 영화를 추천

  => 하지만 평점이 0이거나 낮선 영화들이 있어 수정이 필요
  - ✓ 영화의 평점 정보인 'vote_average' 값을 이용 (소수의 평점 부여는 왜곡됨)

  => 평점을 이용해 필터링해서 최종 추천을 하려고 하는데 평가한 관객이 너무 작은 영화들이 많음
  - ✓ 이와 같은 왜곡된 평점 데이터를 수정하기 위해 '가중치가 부여된 평점' 방식을 이용

1. 추천시스템

❖컨텐츠 기반 필터링 실습 – TMDB 5000 Movie Dataset

- 장르 콘텐츠 필터링을 이용한 영화 추천
  - ✓ 이와 같은 왜곡된 평점 데이터를 수정하기 위해 '가중치가 부여된 평점' 방식을 이용

가중 평점(Weighted Rating) = **(v/(v+m)) * R + (m/(v+m)) * C**

v : 개별 영화 투표 횟수,
m : 최소 투표 횟수,
R : 개별 평균 평점, C : 전체 평균 평점)

m값은 상위 60%에 해당하는 횟수를 기준

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

- https://grouplens.org/datasets/movielens/latest/ ml-latest-small.zip을 다운로드
- 아이템 기반 최근접 이웃 협업 필터링 실습

  1) 아이템-사용자 행렬 데이터로 변환

  2) 아이템 간 코사인 유사도로 아이템 유사도 산출

  3) 사용자가 관람/구매하지 않은 아이템 중 유사도 반영한 예측 점수 계산

  4) 예측 점수가 높은 순으로 아이템 추천

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

- ratings를 이용해 아이템 기반의 최근접 이웃 협업 필터링을 구현할 것인데, 우선 로우 레벨 형태의 원본 데이터를 사용자 로우, 영화 칼럼으로 구성된 사용자-영화 평점 데이터 세트로 변경해야 함(pivot_table() 함수를 이용)

로우 레벨 형태의 사용자 -영화 평점 데이터

userId	movieId	rating
User 1	Movie1	3
User 1	Movie3	3
User 1	Movie6	4
User 2	Movie2	1
User 2	Movie4	5
User 3	Movie5	4

변환

사용자 로우, 영화 칼럼으로 구성된
사용자 -영화 평점 데이터

	Movie 1	Movie 2	Movie 3	Movie 4	Movie 5	Movie 6
User 1	3		3			4
User 2		1		5		
User 3					4	

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

- 영화간 유사도 산출
  - ✓ 이제 사용자-영화 평점 행렬 데이터 세트를 이용해 영화간의 유사도를 측정
  - ✓ 그런데 위 데이터프레임은 userid가 기준인 행 레벨 데이터이므로 유사도를 바로 측정하면 사용자 간의 유사도가 만들어진다. 그러므로 행과 열 위치를 변경해 계산해야 함

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

➢ 아이템 기반 최근접 이웃 협업 필터링으로 개인화된 영화추천

✓ 개인에게 특화된 영화 추천 알고리즘

✓ 이 추천의 가장 큰 특징은 '개인이 관람하지 않은 영화를 추천하는 것'

✓ 그러므로 유사도, 관람한 영화 평점 데이터를 기반으로 해 새롭게 모든 영화의 예측 평점을 계산 후 높은 예측 평점을 가진 영화를 추천하는 방식을 사용

$$\hat{R}_{u,i} = \sum^{N} (S_{i,N} * R_{u,N}) / \sum^{N} (|S_{i,N}|)$$

^Ru,i : 사용자 u, 아이템 i의 개인화된 예측 평점 값
Si,N : 아이템 i와 유사도가 높은 아이템의 유사도 벡터
Ru,N : 유사도가 높은 아이템의 실제 평점 벡터

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

➢ 아이템 기반 최근접 이웃 협업 필터링으로 개인화된 영화추천

✓ 사용자-영화 평점 데이터 프레임을 이용해 사용자별로 최적화된 평점 스코어를 예측하는 함수를 만듬

✓ 이를 이용해 개인화된 예측 평점을 구함

=> 예측 평점이 실제 평점에 비해 작을 수 있는데 , 이는 내적 결과를 코사인 유사도 벡터 합으로 나누었기 때문에 생기는 현상

✓ MSE를 통해 예측 결과가 실제 평점과 얼마나 차이나는지 확인

✓ 주의할 점은 개인화된 예측 점수는 평점을 주지 않은 영화에 대해서도 유사도에 기반해 평점을 예측했으므로 기존에 평점이 부여된 데이터에 대해서만 오차 측정

1. 추천시스템

❖아이템 기반 최근접 이웃 협업 필터링 실습

➢ 아이템 기반 최근접 이웃 협업 필터링으로 개인화된 영화추천

✓ MSE를 감소시키기 위해 특정 영화와 가장 비슷한 유사도를 가지는 영화에 대해서만 유사도 벡터를 적용하는 함수로 변경

=> MSE가 3.69로 기존보다 많이 향상

✓ 이제 특정 사용자에 대해 영화를 추천

✓ 사용자가 평점을 준 영화를 평점이 높은 순으로 나열

✓ 평점을 주지 않은 영화를 리스트 객체로 반환하는 함수 생성

✓ 위의 함수에서 추출한 사용자별 아이템 유사도에 기반한 예측 평점 데이터 세트를 이용해 최종적으로 사용자에게 영화를 추천하는 함수

1. 추천시스템

❖행렬 분해를 이용한 잠재 요인 협업 필터링 실습

- 행렬 분해에는 SVD가 주로 이용되지만 사용자-아이템 평점 행렬에는 NULL 데이터가 많기 때문에 SGD나 AID를 주로 이용
- 이번에는 SGD 기반의 행렬 분해를 구현하고 이를 기반으로 추천

- 확률적 경사 하강법을 이용한 행렬 분해 함수를 이용
- 새롭게 데이터 프레임을 로딩해 다시 사용자-아이템 평점 행렬로 만듬
- 이를 matrix_factorization() 함수를 통해 행렬 분해
- 더 쉽게 영화 아이템 칼럼을 이해하기 위해 반환된 예측 사용자-아이템 평점 행렬을 영화 타이틀을 칼럼명으로 가지는 데이터프레임으로 변경
- 이제 이 행렬 정보를 이용해 개인화된 영화 추천
- 잠재 요인 협업 필터링을 이용해 추천하기 위한 함수를 만듬

1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

- (pip install scikit-surprise) or (conda install -c conda-forge scikit-surprise)
- Surprise 를 이용한 추천시스템 구축
  - ✓ 예제는 추천 데이터를 학습용과 테스트용 데이터세트로 분리한 뒤 SVD 행렬 분해를 통한 잠재 요인 협업 필터링을 수행
  - ✓ 데이터 로딩은 Dataset 클래스를 이용해서만 가능
    - userId/movieId/rating 와 같은 row 레벨 형태로 되어 있는 포맷의 데이터만 처리
    - Dataset 클라스의 load_bulletin() 은 아카이브 사이트로부터 내려받아 로컬 디렉토리에 저장한 뒤 데이터를 로딩

```
data = Dataset.load_builtin('ml-100k')
trainset, testset = train_test_split(data, test_size=.25, random_state=0)
```


1. 추천시스템

❖accuracy.rmse(predictions)

```
algo = SVD()
algo.fit(trainset)
```

```
predictions = algo.test( testset )
print('prediction type :',type(predictions), ' size:',len(predictions))
print('prediction 결과의 최초 5개 추출')
predictions[:5]
```

```
[ (pred.uid, pred.iid, pred.est) for pred in predictions[:3] ]
```

```
# 사용자 아이디, 아이템 아이디는 문자열로 입력해야 함.
uid = str(196)
iid = str(302)
pred = algo.predict(uid, iid)
print(pred)
```

```
accuracy.rmse(predictions)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 주요 모듈 소개

API 명	내용
Dataset.load_builtin (name='ml-100k')	무비렌즈 아카이브 FTP 서버에서 무비렌즈 데이터를 내려받습니다. ml-100k, ml-1M를 내려받을 수 있습니다. 일단 내려받은 데이터는 .surprise_data 디렉터리 밑에 저장되고, 해당 디렉터리에 데이터가 있으면 FTP에서 내려받지 않고 해당 데이터를 이용합니다. 입력 파라미터인 name으로 대상 데이터가 ml-100k인지 ml-1m인지를 입력합니다(name='ml-100k'). 디폴트는 ml-100k입니다.
Dataset.load_from_file (file_path, reader)	OS 파일에서 데이터를 로딩할 때 사용합니다. 콤마, 탭 등으로 칼럼이 분리된 포맷의 OS 파일에서 데이터를 로딩합니다. 입력 파라미터로 OS 파일명, Reader로 파일의 포맷을 지정합니다.
Dataset.load_from_df (df, reader)	판다스의 DataFrame에서 데이터를 로딩합니다. 파라미터로 DataFrame을 입력받으며 DataFrame 역시 반드시 3개의 칼럼인 사용자 아이디, 아이템 아이디, 평점 순으로 칼럼 순서가 정해져 있어야 합니다. 입력 파라미터로 DataFrame 객체, Reader로 파일의 포맷을 지정합니다.

1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 주요 모듈 소개

✓ OS 파일데이터를 Surprise 데이터세트로 로딩

```
import pandas as pd
ratings = pd.read_csv('train/ml-latest-small/ratings.csv')
# ratings_noh.csv 파일로 unload 시 index 와 header를 모두 제거한 새로운 파일 생성.
ratings.to_csv('train/ml-latest-small/ratings_noh.csv', index=False, header=False)
```

```
from surprise import Reader
reader = Reader(line_format='user item rating timestamp', sep=',', rating_scale=(0.5, 5))
data=Dataset.load_from_file('train/ml-latest-small/ratings_noh.csv',reader=reader)
```

```
Trainset, testset = train_test_split(data, test_size=.25, random_state=0)
# 수행시마다 동일한 결과 도출을 위해 random_state 설정
Algo = SVD(n_factors=50, random_state=0)
# 학습 데이터 세트로 학습 후 테스트 데이터 세트로 평점 예측 후 RMSE 평가
algo.fit(trainset)
predictions = algo.test( testset )
accuracy.rmse(predictions)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 주요 모듈 소개

✓ 판다스 DataFrame에서 Surprise 데이터세트 로딩

```
import pandas as pd
from surprise import Reader, Dataset

ratings = pd.read_csv('train/ml-latest-small/ratings.csv')
reader = Reader(rating_scale=(0.5, 5.0))

# ratings DataFrame 에서 컬럼은 사용자 아이디, 아이템 아이디, 평점 순서를 지켜야 합니다.
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)
trainset, testset = train_test_split(data, test_size=.25, random_state=0)

algo = SVD(n_factors=50, random_state=0)
algo.fit(trainset)
predictions = algo.test( testset )
accuracy.rmse(predictions)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 추천 알고리즘 클래스

클래스명	설명
SVD	행렬 분해를 통한 잠재 요인 협업 필터링을 위한 SVD 알고리즘.
KNNBasic	최근접 이웃 협업 필터링을 위한 KNN 알고리즘.
BaselineOnly	사용자 Bias와 아이템 Bias를 감안한 SGD 베이스라인 알고리즘.

파라미터명	내용
n_factors	잠재 요인 K의 개수. 디폴트는 100, 커질수록 정확도가 높아질 수 있으나 과적합 문제가 발생할 수 있습니다.
n_epochs	SGD(Stochastic Gradient Descent) 수행 시 반복 횟수, 디폴트는 20.
biased (bool)	베이스라인 사용자 편향 적용 여부이며, 디폴트는 True입니다.

1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ 베이스라인 평점

✓ 개인의 성향을 반영해 아이템 평가에 편향성 요소를 반영하여 평점을 부과하는 것

모든 사용자의 평균 영화 평점 : 3.5

난 진정한 영화 매니아.
영화 평가는 언제나 깐깐하게

사용자 A 평균 평점
3.0

어벤저스 3편 평균 평점
4.2

사용자 A의 어벤저스 3편 베이스 라인 평점 = 3.5 − 0.5 + 0.7 = 3.7

모든 사용자의 평균 영화 평점		사용자 편향 점수		아이템 편향 점수
3.5	+	3.0 − 3.5 = −0.5	+	4.2 − 3.5 = 0.7

JEONJU UNIVERSITY

1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ 교차 검증(Cross Validation)과 하이퍼 파라미터 튜닝

```
from surprise.model_selection import cross_validate

# Pandas DataFrame에서 Surprise Dataset으로 데이터 로딩
ratings = pd.read_csv('train/ml-latest-small/ratings.csv') # reading data in
pandas df
reader = Reader(rating_scale=(0.5, 5.0))
data = Dataset.load_from_df(ratings[['userId', 'movieId', 'rating']], reader)

algo = SVD(random_state=0)
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose=True)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

- ➢ 교차 검증(Cross Validation)과 하이퍼 파라미터 튜닝

```
from surprise.model_selection import GridSearchCV

# 최적화할 파라미터들을 딕셔너리 형태로 지정.
param_grid = {'n_epochs': [20, 40, 60], 'n_factors': [50, 100, 200] }

# CV를 3개 폴드 세트로 지정, 성능 평가는 rmse, mse 로 수행 하도록 GridSearchCV 구성
gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=3)
gs.fit(data)

# 최고 RMSE Evaluation 점수와 그때의 하이퍼 파라미터
print(gs.best_score['rmse'])
print(gs.best_params['rmse'])
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 를 이용한 개인화 영화 추천 시스템 구축

```
from surprise.dataset import DatasetAutoFolds

reader = Reader(line_format='user item rating timestamp', sep=',', rating_scale=(0.5, 5))
# DatasetAutoFolds 클래스를 ratings_noh.csv 파일 기반으로 생성.
data_folds = DatasetAutoFolds(ratings_file='train/ml-latest-small/ratings_noh.csv',
reader=reader)

#전체 데이터를 학습데이터로 생성함.
trainset = data_folds.build_full_trainset()
```

```
algo = SVD(n_epochs=20, n_factors=50, random_state=0)
algo.fit(trainset)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 를 이용한 개인화 영화 추천 시스템 구축

```
# 영화에 대한 상세 속성 정보 DataFrame로딩
movies = pd.read_csv('train/ml-latest-small/movies.csv')

# userId=9 의 movieId 데이터 추출하여 movieId=42 데이터가 있는지 확인.
movieIds = ratings[ratings['userId']==9]['movieId']
if movieIds[movieIds==42].count() == 0:
   print('사용자 아이디 9는 영화 아이디 42의 평점 없음')

print(movies[movies['movieId']==42])
```

```
uid = str(9)
iid = str(42)

pred = algo.predict(uid, iid, verbose=True)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 를 이용한 개인화 영화 추천 시스템 구축

```
def get_unseen_surprise(ratings, movies, userId):
    #입력값으로 들어온 userId에 해당하는 사용자가 평점을 매긴 모든 영화를 리스트로 생성
    seen_movies = ratings[ratings['userId']== userId]['movieId'].tolist()

    # 모든 영화들의 movieId를 리스트로 생성.
    total_movies = movies['movieId'].tolist()

    # 모든 영화들의 movieId중 이미 평점을 매긴 영화의 movieId를 제외하여 리스트로 생성
    unseen_movies= [movie for movie in total_movies if movie not in seen_movies]
    print('평점 매긴 영화수:',len(seen_movies), '추천대상 영화수:',len(unseen_movies), \
        '전체 영화수:',len(total_movies))

    return unseen_movies

unseen_movies = get_unseen_surprise(ratings, movies, 9)
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

➢ Surprise 를 이용한 개인화 영화 추천 시스템 구축

```
def recomm_movie_by_surprise(algo, userId, unseen_movies, top_n=10):
    # 알고리즘 객체의 predict() 메서드를 평점이 없는 영화에 반복 수행한 후 결과를 list 객체로 저장
    predictions = [algo.predict(str(userId), str(movieId)) for movieId in unseen_movies]
        # predictions list 객체는 surprise의 Predictions 객체를 원소로 가지고 있음.
    # [Prediction(uid='9', iid='1', est=3.69), Prediction(uid='9', iid='2', est=2.98),,,,]
    # 이를 est 값으로 정렬하기 위해서 아래의 sortkey_est 함수를 정의함.
    # sortkey_est 함수는 list 객체의 sort() 함수의 키 값으로 사용되어 정렬 수행.
    def sortkey_est(pred):
        return pred.est
        # sortkey_est( ) 반환값의 내림 차순으로 정렬 수행하고 top_n개의 최상위 값 추출.
    predictions.sort(key=sortkey_est, reverse=True)
    top_predictions= predictions[:top_n]
        # top_n으로 추출된 영화의 정보 추출. 영화 아이디, 추천 예상 평점, 제목 추출
    top_movie_ids = [ int(pred.iid) for pred in top_predictions]
    top_movie_rating = [ pred.est for pred in top_predictions]
    top_movie_titles = movies[movies.movieId.isin(top_movie_ids)]['title']
    top_movie_preds = [ (id, title, rating) for id, title, rating in zip(top_movie_ids, top_movie_titles,
top_movie_rating)]
        return top_movie_preds
```


1. 추천시스템

❖파이썬 추천시스템 패키지 - Surprise

- Surprise 를 이용한 개인화 영화 추천 시스템 구축

```
unseen_movies = get_unseen_surprise(ratings, movies, 9)
top_movie_preds = recomm_movie_by_surprise(algo, 9, unseen_movies, top_n=10)
print('##### Top-10 추천 영화 리스트 #####')
for top_movie in top_movie_preds:
    print(top_movie[1], ":", top_movie[2])
```