Introduction to Recommender Systems

박치완 <chiwanpark@hotmail.com>

2019년 10월

튜토리얼 내용

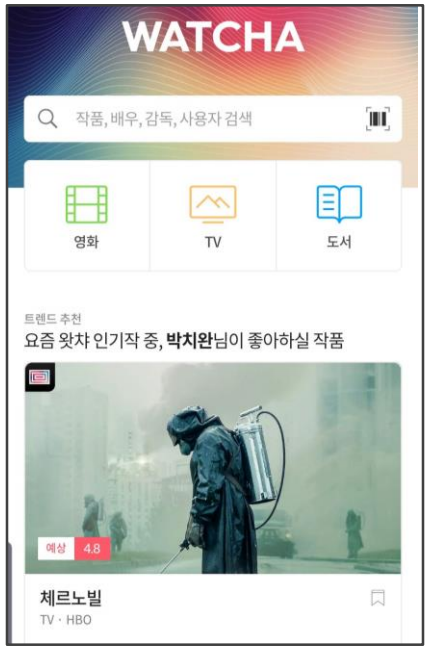
- 튜토리얼 목표
- 기초 이론
  - 추천 시스템 정의
  - 다양한 추천 시스템 분류
  - 다양한 방식의 추천 알고리즘
  - 추천 알고리즘 평가 방법
- 심화 이론 + 실습
  - 다음 영화 평가 데이터 소개
  - 간단한 영화 추천 시스템 만들어 보기

튜토리얼 목표

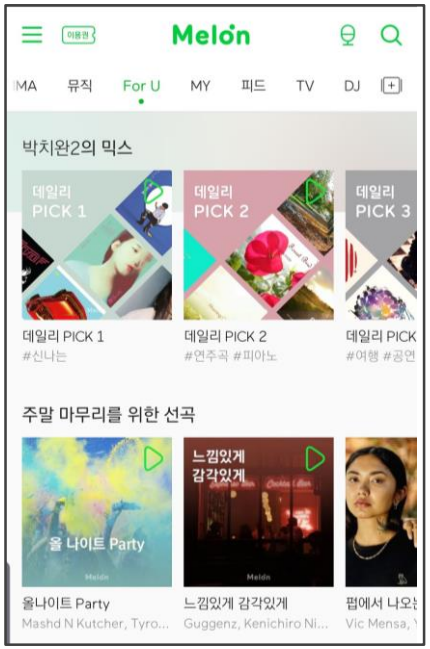
1. 추천 분야에 대한 기초적인 지식 전달
   - 추천 알고리즘의 분류 및 평가 방법에 대한 지식
   - 추천 분야에서 데이터 분석가의 역할
2. 간단한 영화 추천 시스템 만들어 보기

추천 시스템

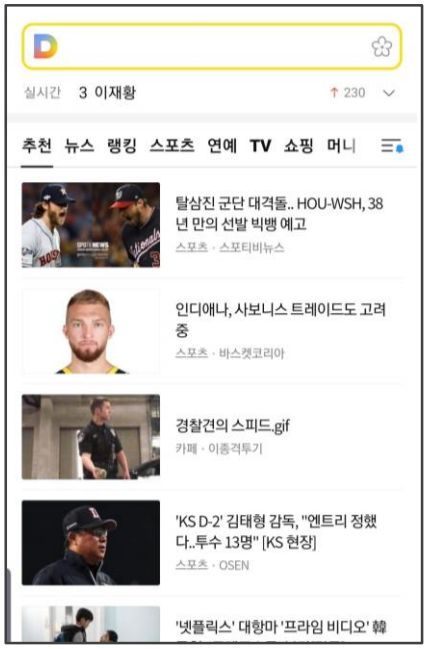
- 주변에서 자주 접하는 추천 시스템 예시



왓챠



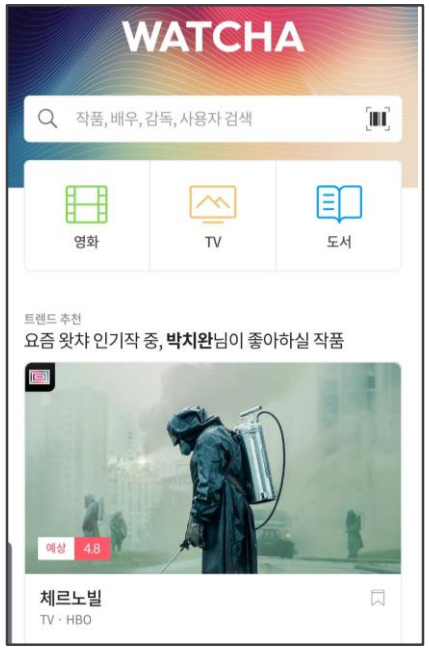
멜론



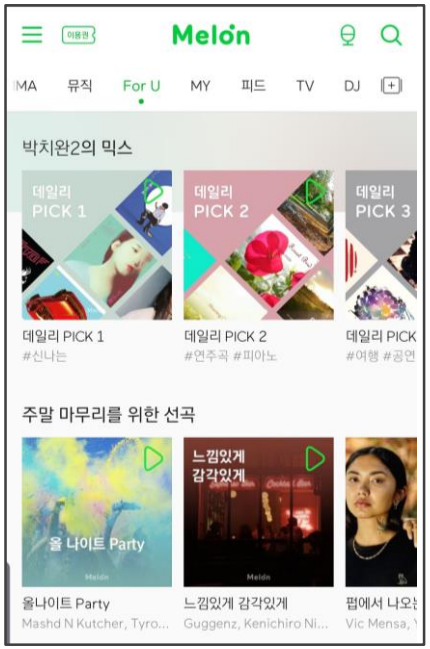
Daum 앱

추천 시스템

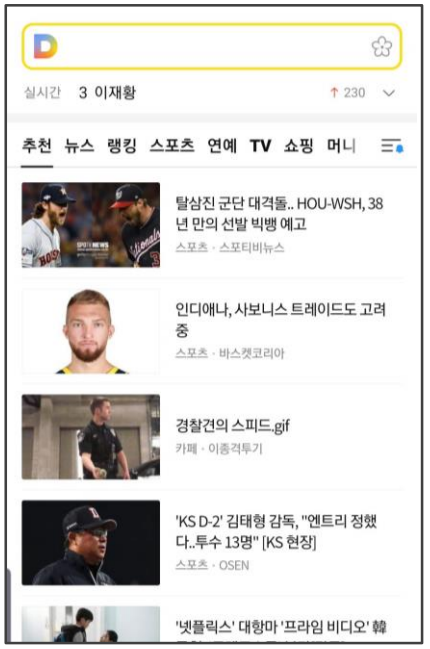
- 이 서비스들의 공통점은 무엇일까요?



왓챠



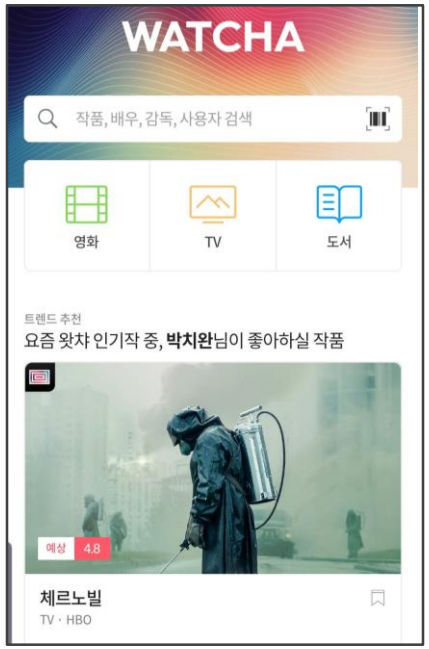
멜론



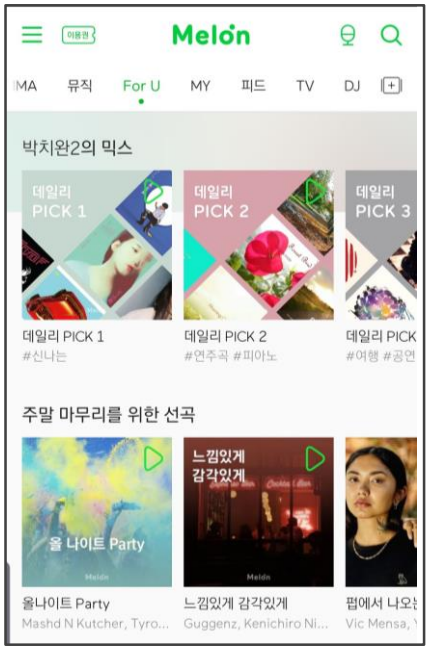
Daum 앱

추천 시스템

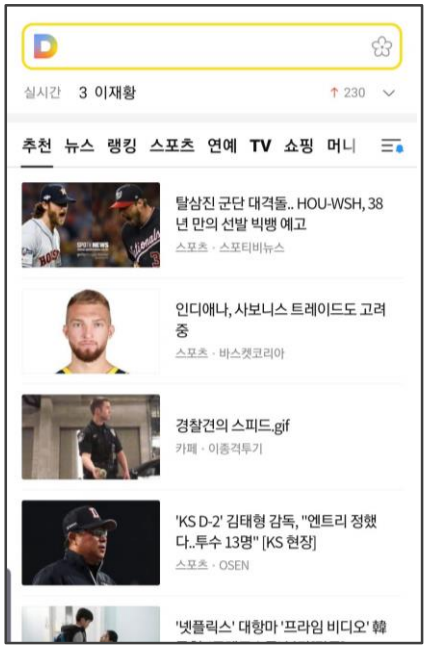
- 이 서비스들의 공통점은 무엇일까요?



왓챠
영화 수 만편



멜론
음악 수십 만 곡



Daum 앱
컨텐츠 수백 만 개

정보 필터링

- 정보의 양이 폭증함에 따라 정보 소비자가 "원하는" 정보를 얻는데 시간과 노력이 많이 필요
- 정보 소비자에게 원하는 정보를 쉽게 얻도록 도와주는 분야
- 정보 필터링의 대표적인 분야
  - 검색
  - 추천 시스템

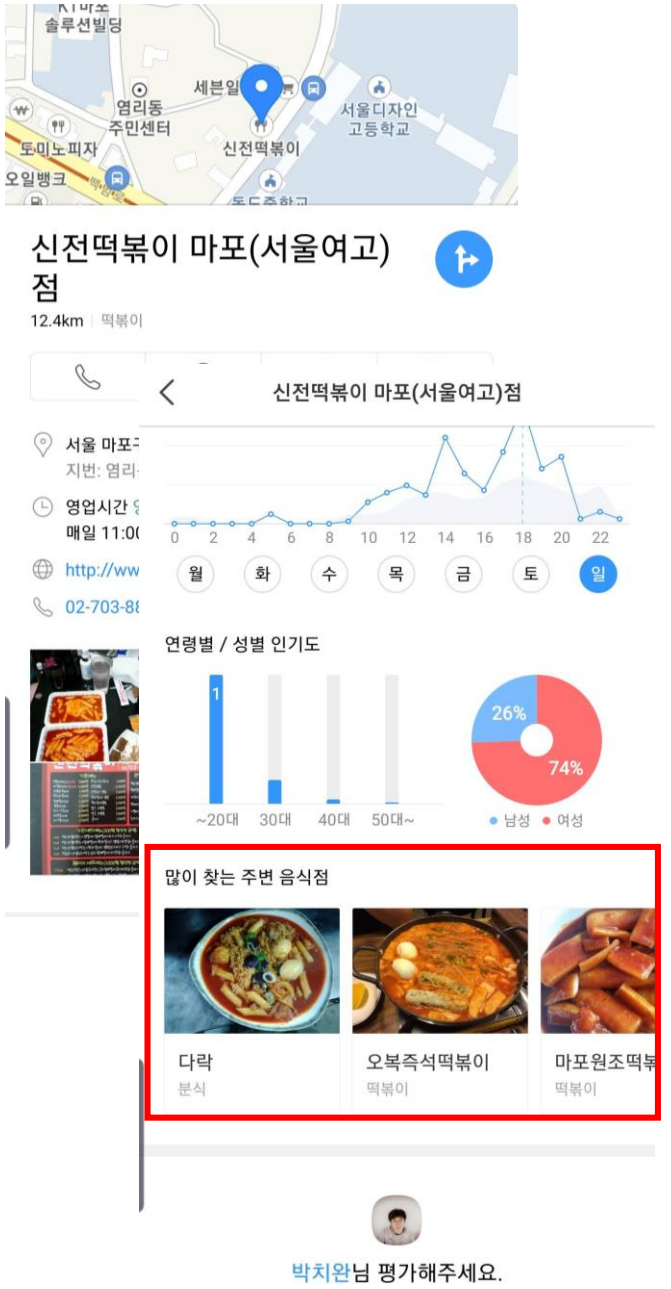
추천 시스템

- 정보 소비자가 "원하는" 정보를 찾아 소비자에게 추천하는 시스템

- 검색과의 차이점
  - 검색은 소비자가 관심사를 표현하는 "검색"이라는 행위를 해야함 (active)
  - 추천은 특별한 행위 없이도 정보 전달이 가능 (passive)

추천 시스템 분류

- 시나리오에 따른 분류
  - 연관된 아이템 추천
    - 현재 소비되고 있는 아이템과 연관된 아이템을 추천



추천 시스템 분류

- 시나리오에 따른 분류
  - 개인화 아이템 추천
    - 소비중인 아이템이 없더라도, 개인의 관심사를 찾아 소비할 만한 아이템을 추천

추천 시스템 분류

- 피드백 종류에 따른 분류
  - 명시적 피드백을 사용하는 추천 시스템
    - 영화 평점 / 좋아요, 싫어요와 같이 소비자가 "명시적으로" 자신의 선호를 표현한 데이터 = 명시적 피드백
  - 암시적 피드백을 사용하는 추천 시스템
    - 웹 페이지 접속 기록 (상품), 음악 청취 기록 같이 소비자가 명시적으로 표현하진 않은 피드백 = 암시적 피드백

추천 시스템 분류

- 업데이트 주기에 따른 분류
  - 정적 추천 시스템
    - 특정 시점의 데이터를 사용해 추천 결과를 계산하는 추천 방식
  - 동적 추천 시스템
    - 지속적으로 사용자의 데이터를 받아 추천 결과를 업데이트하는 추천 방식

추천 시스템 분류

- 시나리오에 따른 분류
  - 연관된 아이템 추천
  - 개인화 아이템 추천
- 피드백 종류에 따른 분류
  - 명시적 피드백을 사용하는 추천 시스템
  - 암시적 피드백을 사용하는 추천 시스템
- 업데이트 주기에 따른 분류
  - 정적 추천 시스템
  - 동적 추천 시스템

- 실제로는 다양한 추천 로직이 섞인 하이브리드 추천 시스템을 많이 사용

추천 알고리즘

- Knowledge-based Filtering
  - 추천하고자 하는 분야의 도메인 지식을 활용해 추천하는 방식
  - e.g. 성별/연령별로 많이 팔리는 상품들을 모아 추천에 활용

추천 알고리즘

- Content-based Filtering
  - 추천 하려는 아이템의 콘텐츠 정보를 분석하거나 정리된 메타 정보를 활용해 콘텐츠별로 특징 정보를 만들고 이를 활용해 추천
  - e.g. 액션 장르 영화 페이지 하단에 다른 액션 장르의 영화를 추천

추천 알고리즘

- Collaborative Filtering
  - 소비자의 아이템 소비 이력을 사용해 소비하지 않은 새로운 아이템을 추천
  - e.g. 영화 감상 이력을 바탕으로 소비자가 좋아할 만한 영화를 찾아내서 추천

추천 알고리즘 평가 방법

- 오프라인 평가 (Offline Evaluation)
  - 사용자의 아이템에 대한 선호 기록과 추천 시스템이 추천한 결과를 비교하여 추천 품질을 평가

  - 특징
    - 별다른 비용 지출 없이 수집된 데이터만 사용하여 평가가 가능
    - 여러 모델을 동시에 평가할 수 있음
    - 선호 기록이 기존에 사용 중인 추천 모델에 영향을 받을 수 있으므로 실제 사용자의 만족도와 평가 결과가 다를 수 있음

추천 알고리즘 평가 방법

- 온라인 평가 (Online Evaluation)
  - 만들어진 추천 시스템을 직접 사용자에게 노출시켜 사용자의 반응을 수집하여 추천 품질을 평가
    - A/B 테스트나 통계적인 샘플링을 적용
  - 특징
    - 시간의 흐름에 따른 평가가 가능하고 실제 사용자의 만족도를 측정한다는 측면에서 정확한 방식
    - 비용이 비싼 평가 방식 (사용자의 만족도 감소 가능성 등)

다음 영화 평가 데이터

- 다음 영화(https://movie.daum.net) 서비스를 크롤링 한 데이터
- 데이터 (CSV 포맷)
  - https://bit.ly/recsys-daum-movie
- 데이터를 받아 여러분 컴퓨터에 저장해 주세요

다음 영화 평가 데이터

- 지금부터 여러분이 해야하는 일
  1. 영화 메타데이터(metadata.csv)를 pandas DataFrame으로 불러오기
  2. 평점 데이터(ratings-train.csv)를 pandas DataFrame으로 불러오기
  3. 평점의 분포를 히스토그램으로 그려보기
  4. 이외의 EDA를 통해 데이터의 패턴을 발견해보기

영화 추천 시스템

- 영화 추천 문제
  - 소비자 u가 평점을 남기지 않은 영화 i에 대해 예측 평점 $\widehat{r_{u,i}}$를 계산하는 문제
    - 예측 평점을 계산할 수 있다면, 예측 평점이 높은 순으로 추천해주면 추천 시스템 완성!

```
def predict(u, i):
    return <expected rating of i given user u>
```

- 지금부터 다양한 방법으로 predict 함수를 구현해 볼 예정

영화 추천 시스템

- 평가 방법
  - 오프라인 평가
  - RMSE (Root Mean Square Error)
    - $RMSE = \sqrt{\frac{1}{|D|}\sum_{u,i \in D}\left(r_{u,i} - \widehat{r_{u,i}}\right)^2}$
    - $r_{u,i}$ = 사용자 u가 영화 i에 부여한 실제 평점
    - 낮을수록 좋은 점수 (error)
  - 미리 준비된 validation 데이터(ratings-valid.csv)를 사용해 측정
    - Validation 데이터도 DataFrame으로 불러와주세요

영화 추천 시스템

- RMSE를 계산하는 rmse 함수를 만들어봅시다

```
def rmse(expected, answer):
    return <root mean square error between expected and answer>
```

  - expected: 여러분이 계산한 (userid, itemid, rating)이 들어있는 DataFrame
  - answer: 정답 (userid, itemid, rating)이 들어있는 DataFrame
  - Hint: pandas outer join

영화 추천 시스템 만들기 (1)

1. 예측 평점을 전체 데이터의 평균 평점으로 계산하는 추천 시스템 만들어 보기
   - predict 함수에서 어떻게 값을 돌려줘야 하는지 생각해 보세요
2. RMSE 얼마 나오는지 확인해보기

영화 추천 시스템 만들기 (2)

1. 평점을 각 소비자의 평균 평점으로 예측하는 추천 시스템 만들어보기
2. 평점을 각 영화의 평균 평점으로 예측하는 추천 시스템 만들어보기
3. RMSE 측정

영화 추천 시스템 만들기 (3)

- User-User Collaborative Filtering
  - 지금까지 보다 좀 더 진보된 추천 방식
  - 주어진 소비자와 성향이 비슷한 다른 사용자들을 찾아 그 소비자들이 남긴 평점을 바탕으로 평점을 예측

	영화 1	영화 2	영화 3	영화 4
소비자 1	1.0	?	3.5	4.0
소비자 2		2.0	3.0	4.0
소비자 3	4.0		4.0	
소비자 4	2.0			2.5

영화 추천 시스템 만들기 (3)

- User-User Collaborative Filtering
  1. 서로 다른 두 소비자 u와 v에 대해 유사한 정도를 나타내는 $sim(u, v)$ 함수를 정의해야 함
     - 유사도는 여러 가지를 쓸 수 있음 (Jaccard, Cosine)
  2. 주어진 소비자 u에 대해 비슷한 소비자 k명(U_u)을 찾음
  3. 찾은 k명의 소비자가 영화 i에 대해 남긴 가중 평균 평점($\frac{1}{\sum sim(u,v)} \sum_{v \in U_u} sim(u, v) \cdot r_{v,i}$)을 계산하여 소비자 u의 예측 평점($\widehat{r_{u,i}}$)으로 사용

영화 추천 시스템 만들기 (3)

1. 유사도 함수 $sim(u, v)$
   - Jaccard similarity
     - $sim(u, v) = \frac{|I_u \cap I_v|}{|I_u \cup I_v|}$
     - $I_u = \{i \mid \exists r_{u,i}\}$ (소비자 u가 평점을 남긴 영화 집합)
     - $I_v = \{i \mid \exists r_{v,i}\}$ (소비자 v가 평점을 남긴 영화 집합)

```
def sim(u, v):
    return <similarity between users u and v>
```

영화 추천 시스템 만들기 (3)

2. 비슷한 영화를 소비한 k명의 소비자 U_u를 찾는 함수 $similar\_users(u, k)$
   - 모든 유저에 대해 소비자 u와의 유사도를 계산 $(sim(u, v))$
   - 유사도를 정렬한 뒤, 유사도가 큰 k명의 유저만 돌려줌

```
def similar_users(u, k):
    return <top-k users in terms of user similarity>
```

영화 추천 시스템 만들기 (3)

3. 소비자 u와 영화 i에 대한 예측 평점을 계산하는 $predict(u, i)$ 함수
   - $\widehat{r_{u,i}} = \frac{1}{\sum sim(u,v)} \sum_{v \in U_u} sim(u, v) \cdot r_{v,i}$
   - 앞서 만든, $sim(u, v)$ 함수와 $similar\_users(u, k)$ 함수를 적절히 활용

```
def predict(u, i):
    return <expected rating of i given user u>
```

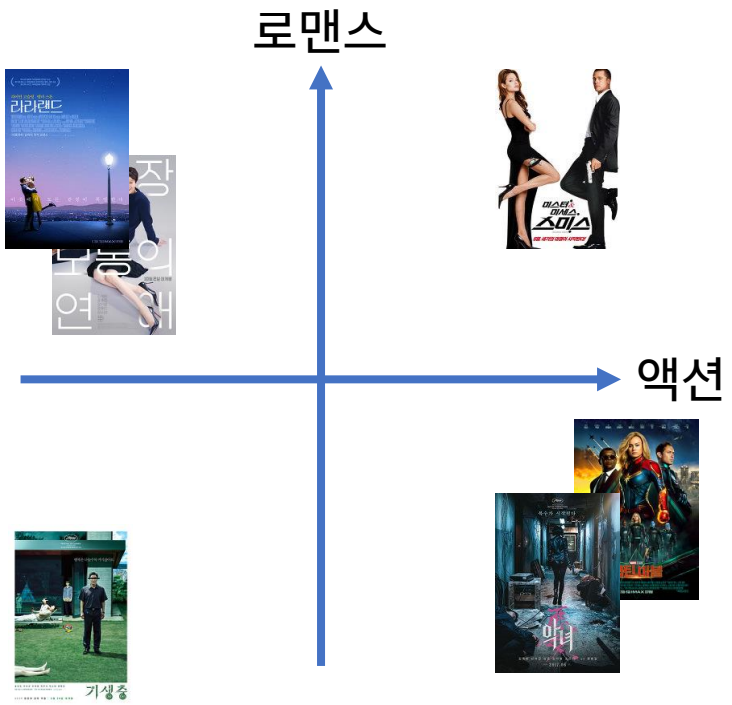
영화 추천 시스템 만들기 (3)

4. 만든 predict 함수를 사용해 예측 평점을 계산하고 RMSE 측정
5. (Bonus) 예측 평점을 아래 처럼 바꿔 계산하는 것이 기존 모델과 어떤 부분에서 다른지 생각해 보기 ($\overline{r_u}$는 사용자 u의 평균 평점)

$$\widehat{r_{u,i}} = \overline{r_u} + \frac{1}{\sum sim(u,v)} \sum_{v \in U_u} sim(u,v)(r_{v,i} - \overline{r_v})$$

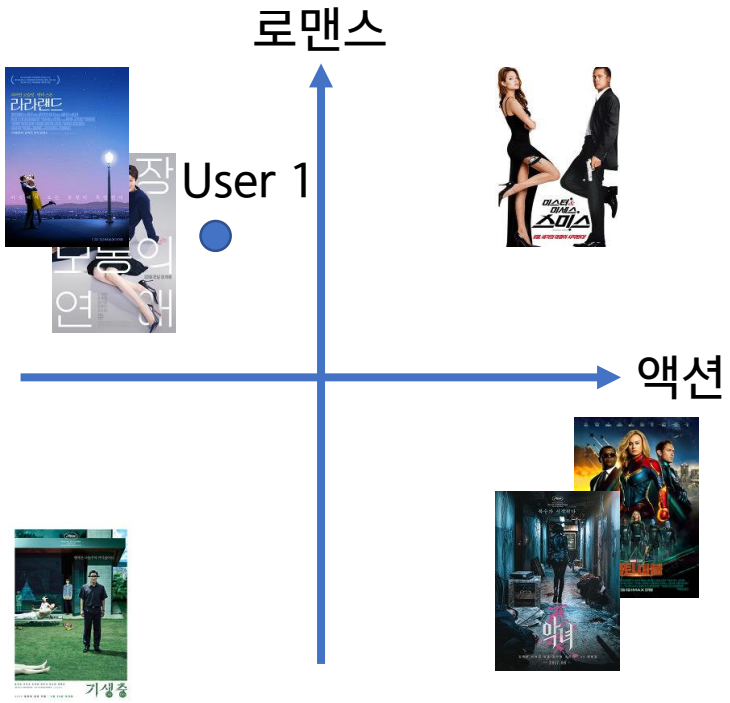
영화 추천 시스템 만들기 (4)

- Latent Factor Model
  - 소비자, 아이템에 각각 잠재적인 특성(latent factor)이 있다고 가정하고 이를 학습하고자 하는 모델
    - Latent factor는 주로 n차원 벡터로 표현



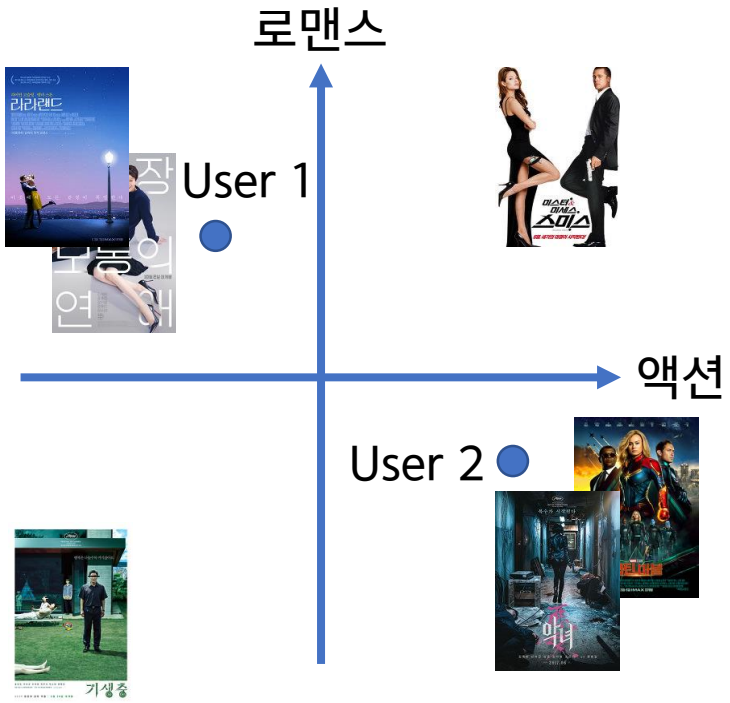
영화 추천 시스템 만들기 (4)

- Latent Factor Model
  - 소비자, 아이템에 각각 잠재적인 특성(latent factor)이 있다고 가정하고 이를 학습하고자 하는 모델
    - Latent factor는 주로 n차원 벡터로 표현



영화 추천 시스템 만들기 (4)

- Latent Factor Model
  - 소비자, 아이템에 각각 잠재적인 특성(latent factor)이 있다고 가정하고 이를 학습하고자 하는 모델
    - Latent factor는 주로 n차원 벡터로 표현



영화 추천 시스템 만들기 (4)

- Matrix Factorization (MF)
  - 소비자의 영화에 대한 평점 데이터는 행렬로 표현할 수 있음

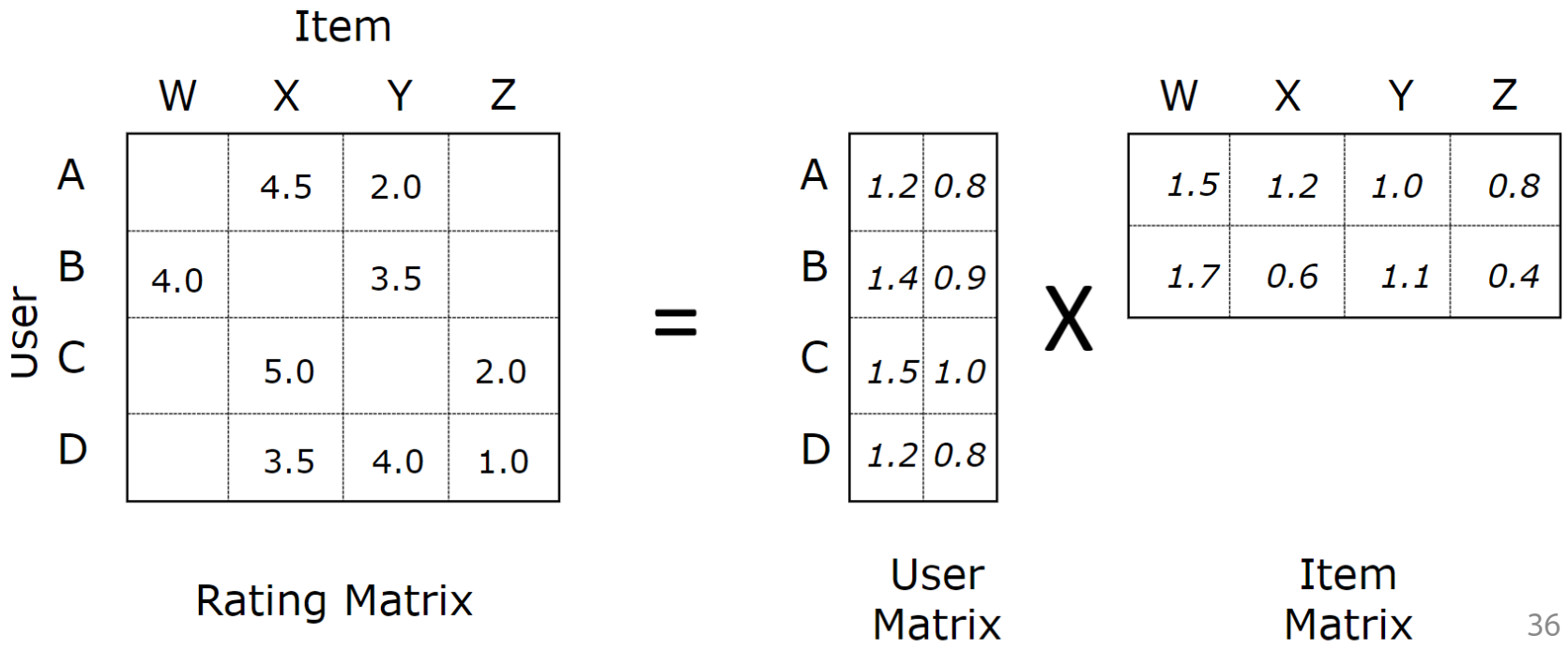
Item

User	W	X	Y	Z
A		4.5	2.0	
B	4.0		3.5	
C		5.0		2.0
D		3.5	4.0	1.0

Rating Matrix

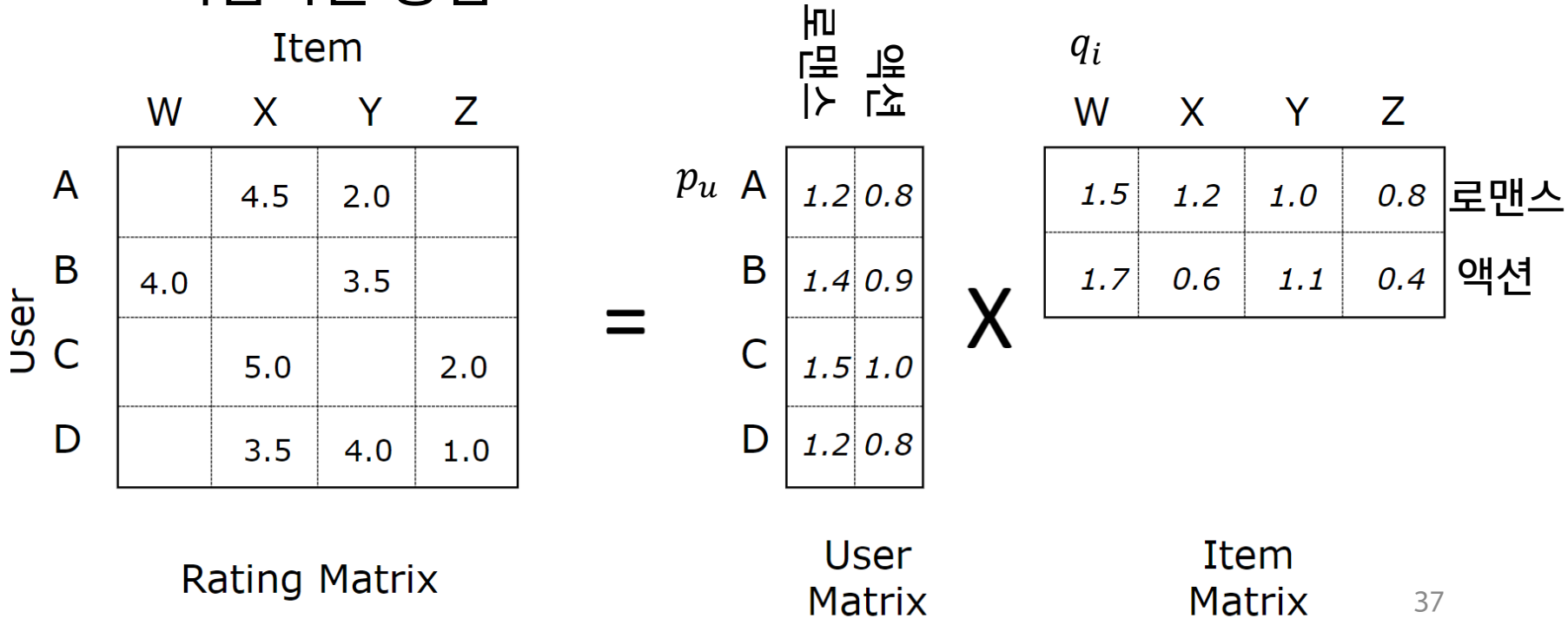
영화 추천 시스템 만들기 (4)

- Matrix Factorization (MF)
  - 표현한 행렬을 서로 다른 두 저차원 행렬의 곱으로 나타낼 수 있도록 (복원할 수 있도록) 저차원 행렬을 학습하는 방법



영화 추천 시스템 만들기 (4)

- Matrix Factorization (MF)
  - 표현한 행렬을 서로 다른 두 저차원 행렬의 곱으로 나타낼 수 있도록 (복원할 수 있도록) 저차원 행렬을 학습하는 방법

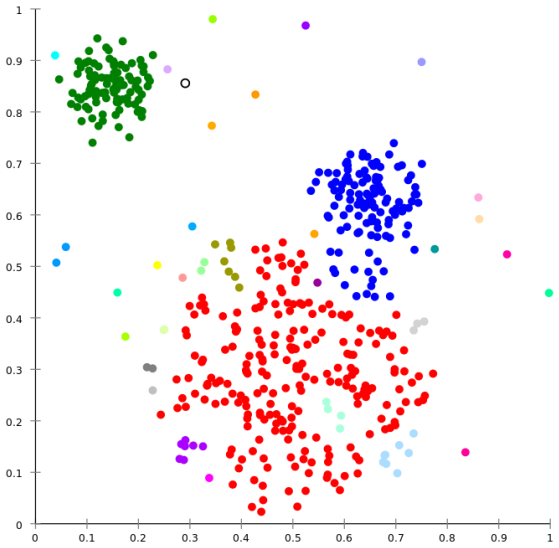


영화 추천 시스템 만들기 (4)

- Matrix Factorization (MF)
  - 로맨스 영화를 좋아하고, 액션 영화를 싫어하는 소비자의 latent factor (user factor / user feature)
    - $p_u = (1.9, -2.4)$
  - 로맨스 영화의 latent factor (item factor / item feature)
    - $q_i = (2.7, -1.9)$
  - 두 latent factor로 부터 계산한 예측 평점
    - $\widehat{r_{u,i}} = p_u \cdot q_i = (1.9, -2.4) \cdot (2.7, -1.9) = 9.69$

영화 추천 시스템 만들기 (4)

- Matrix Factorization의 장점
  - User / Item을 표현하는 저차원(low dimensional) 밀집 벡터(dense vector)를 얻을 수 있어서 다양한 머신 러닝 / 데이터 분석 방법을 적용할 수 있음
    - Cluster analysis, 다른 모델의 입력으로 사용



영화 추천 시스템 만들기 (4)

- Matrix Factorization의 장점
  - 복원하고자 하는 행렬을 어떻게 구성하냐에 따라 다양하게 활용할 수 있음
    - 다양한 워드 임베딩 모델 중 일부는 단어-단어 행렬을 구성하고 matrix factorization을 적용한 것
      - Word2Vec (Mikolov et. al, 2013) - SPPMI
      - GloVe (Pennington et. al, 2014) - Co-occurrence prob.
    - 소셜 네트워크 데이터를 활용해 사용자-사용자 행렬을 구성하고 matrix factorization을 적용해 사용자 특성 추출

영화 추천 시스템 만들기 (4)

- Matrix Factorization (학습)
  - Objective function

$$L = \sum_{u,i \in R} \left(r_{u,i} - \widehat{r_{u,i}}\right)^2$$

$$= \sum_{u,i \in R} \left(r_{u,i} - p_u \cdot q_i\right)^2$$

영화 추천 시스템 만들기 (4)

- Matrix Factorization (학습)
  - Objective function

$$L = \sum_{u,i \in R} \left(r_{u,i} - \widehat{r_{u,i}}\right)^2 + \lambda(\|p_u\|_2^2 + \|q_i\|_2^2)$$

$$= \sum_{u,i \in R} \left(r_{u,i} - p_u \cdot q_i\right)^2 + \lambda(\|p_u\|_2^2 + \|q_i\|_2^2)$$

영화 추천 시스템 만들기 (4)

- Matrix Factorization (학습)
  - Objective function

$$L = \sum_{u,i \in R} \left(r_{u,i} - \widehat{r_{u,i}}\right)^2 + \lambda(\|p_u\|_2^2 + \|q_i\|_2^2)$$

$$= \sum_{u,i \in R} \left(r_{u,i} - p_u \cdot q_i\right)^2 + \lambda(\|p_u\|_2^2 + \|q_i\|_2^2)$$

  - 학습 방법
    - Alternating least squares
    - Gradient descent (or its variant)

영화 추천 시스템 만들기 (4)

- 지금부터 여러분이 해야하는 일
  1. 지난번에 구현했던 User-User CF를 Surprise 라이브러리로 수행해보기 (같이)
  2. Matrix factorization을 Surprise 라이브러리로 수행해보기 (SVD)
  3. Matrix factorization의 다양한 옵션을 문서에서 확인해보고 바꿔서 결과 확인해보기

영화 추천 시스템 만들기 (4)

- Funk's SVD
  - 소비자의 기본 성격과 아이템의 기본 인기도를 bias로 표현하여 모델에 반영

$$L = \sum_{u,i \in R} \left(r_{u,i} - \widehat{r_{u,i}}\right)^2 + \lambda(\|p_u\|_2 + \|q_i\|_2)$$

$$= \sum_{u,i \in R} \left(r_{u,i} - (p_u \cdot q_i + b_u + b_i + \mu)\right)^2$$

$$+ \lambda(b_u^2 + b_i^2 + \|p_u\|_2 + \|q_i\|_2)$$

  - Surprise SVD에서 biased=True (기본값) 옵션을 주면 Funk's SVD를 사용함

유사 엔티티 분석

- MF를 통해 얻은 latent factor로 유사한 소비자나 아이템을 찾아 분석하는 것이 가능
- 일반적으로 matrix factorization으로 학습한 latent factor는 cosine similarity로 유사도를 측정

$$sim(u, v) = \frac{u \cdot v}{\|u\|\|v\|}$$

유사 엔티티 분석

- Surprise 라이브러리의 matrix factorization 모델은 latent factor를 변수에 저장해 줌
  - pu = user latent factor
  - qi = item latent factor
- DataFrame에 들어있던 userid / itemid를 matrix row, column index로 변환해서 접근해야 함

유사 엔티티 분석

- ‘QjEzc3k=‘ 소비자가 10점을 준 영화 목록을 뽑아보고, 해당 소비자와 cosine similarity가 높은 2명이 소비자가 10점을 준 영화 목록을 출력해보기

유사 엔티티 분석

- 시각화 툴을 이용한 유사 엔티티 분석
  - Tensorflow Embedding Projector (https://projector.tensorflow.org)
  - qi.tsv
    - 한 줄에 qi 벡터가 하나씩 출력되어야 함 (tab separated)
  - titles.tsv
    - 한 줄에 영화 제목이 하나씩 출력되어야 함
  - 데이터를 업로드 해 여러 가지 영화의 유사 영화를 확인해보며 분석을 해보기

Summary

- 추천 시스템이 무엇이고 왜 필요한가?
- 추천 시스템의 다양한 분류 및 추천 알고리즘 종류
  - Latent Factor Model
  - Matrix Factorization
- 유사 엔티티 분석은 어떻게 하는가?