비즈니스 Case Study를 통한

추천 시스템 구현

2강

컨텐츠 기반 추천 & 협업 필터링

2강

컨텐츠 기반 추천 & 협업 필터링

Contents

1. 컨텐츠 기반 추천, TF-IDF
2. 협업 필터링의 원리
3. User-based, Item-based
4. 유사도 개념 이해하기

1. 컨텐츠 기반 추천, TF-IDF

Content-based Recommendation

기본 IDEA: 유저 x가 과거에 선호한 아이템과 비슷한 아이템을 유저 x에게 추천

예시

영화 : 배우, 감독, 영화 장르

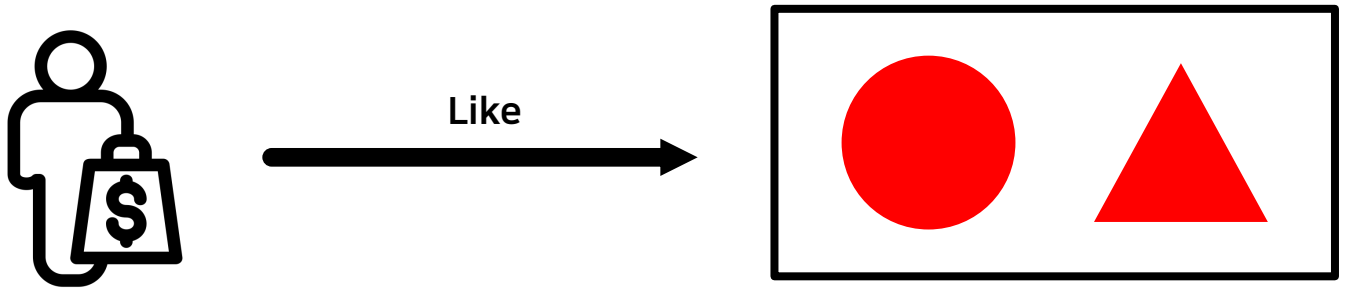
음악 : 아티스트, 장르, 리듬, 무드

블로그 / 뉴스 : 비슷한 주제나 내용을 가진 텍스트(문장, 단어)

사람 : 공통의 친구를 많이 가진 다른 사람

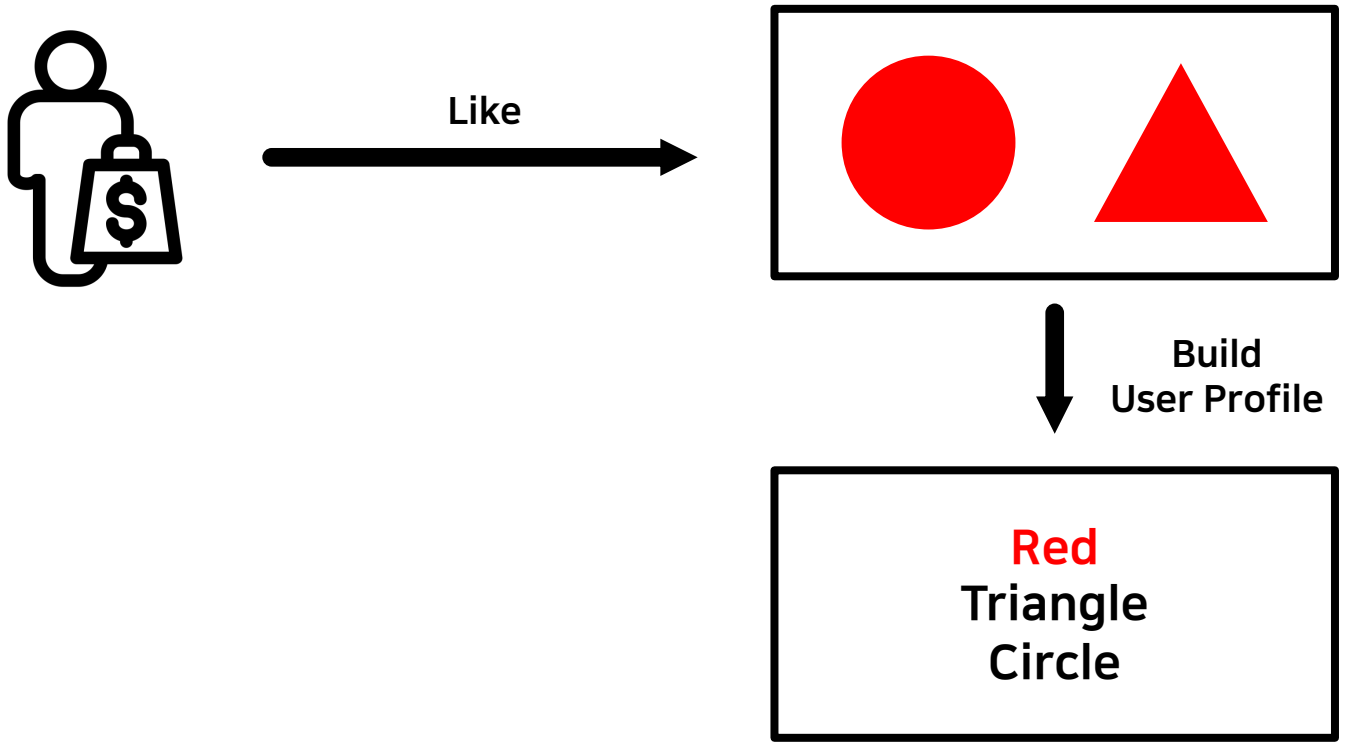
Content-based Recommendation

유저가 선호하는 아이템을 기반으로 해당 아이템과 유사한 아이템을 추천



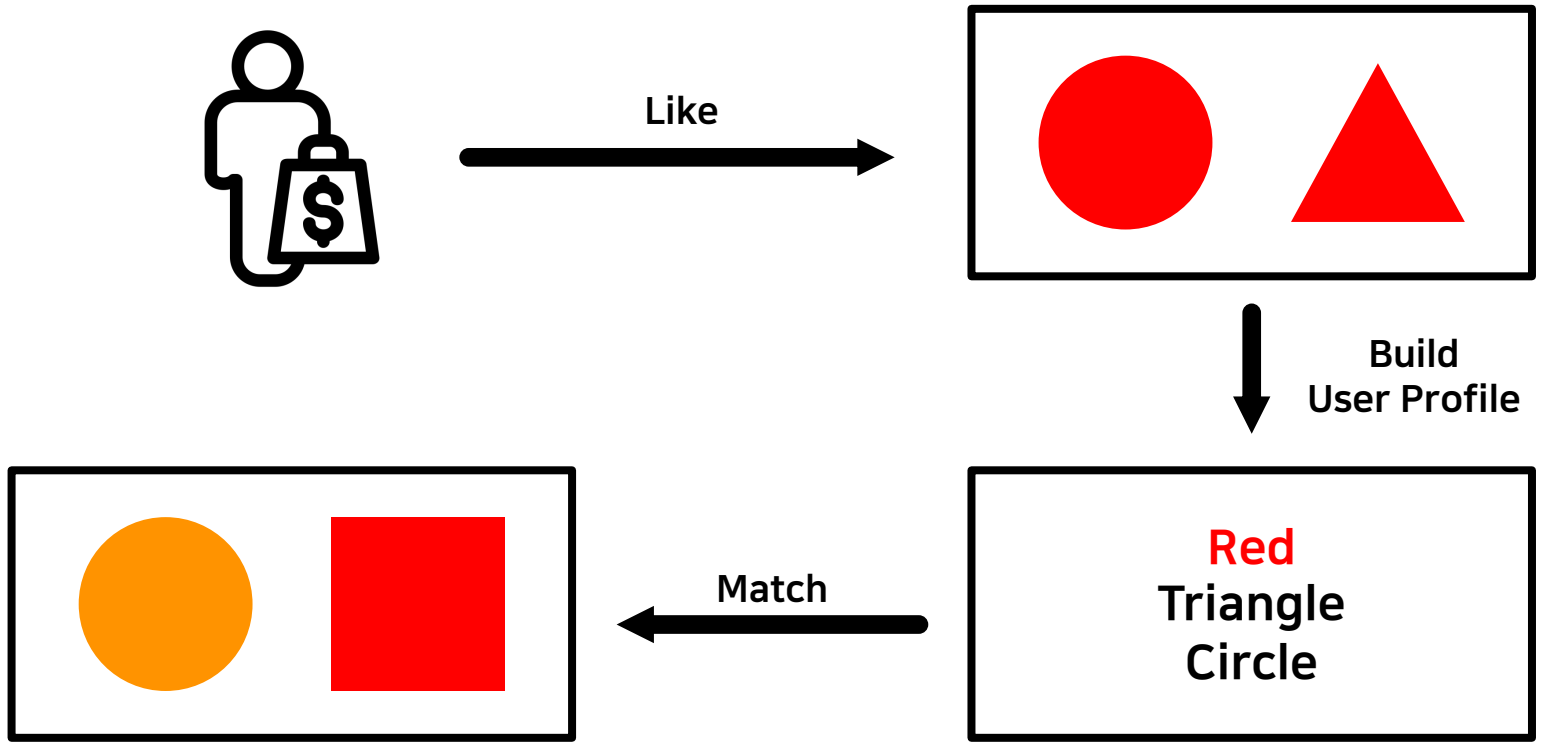
Content-based Recommendation

유저가 선호하는 아이템을 기반으로 해당 아이템과 유사한 아이템을 추천



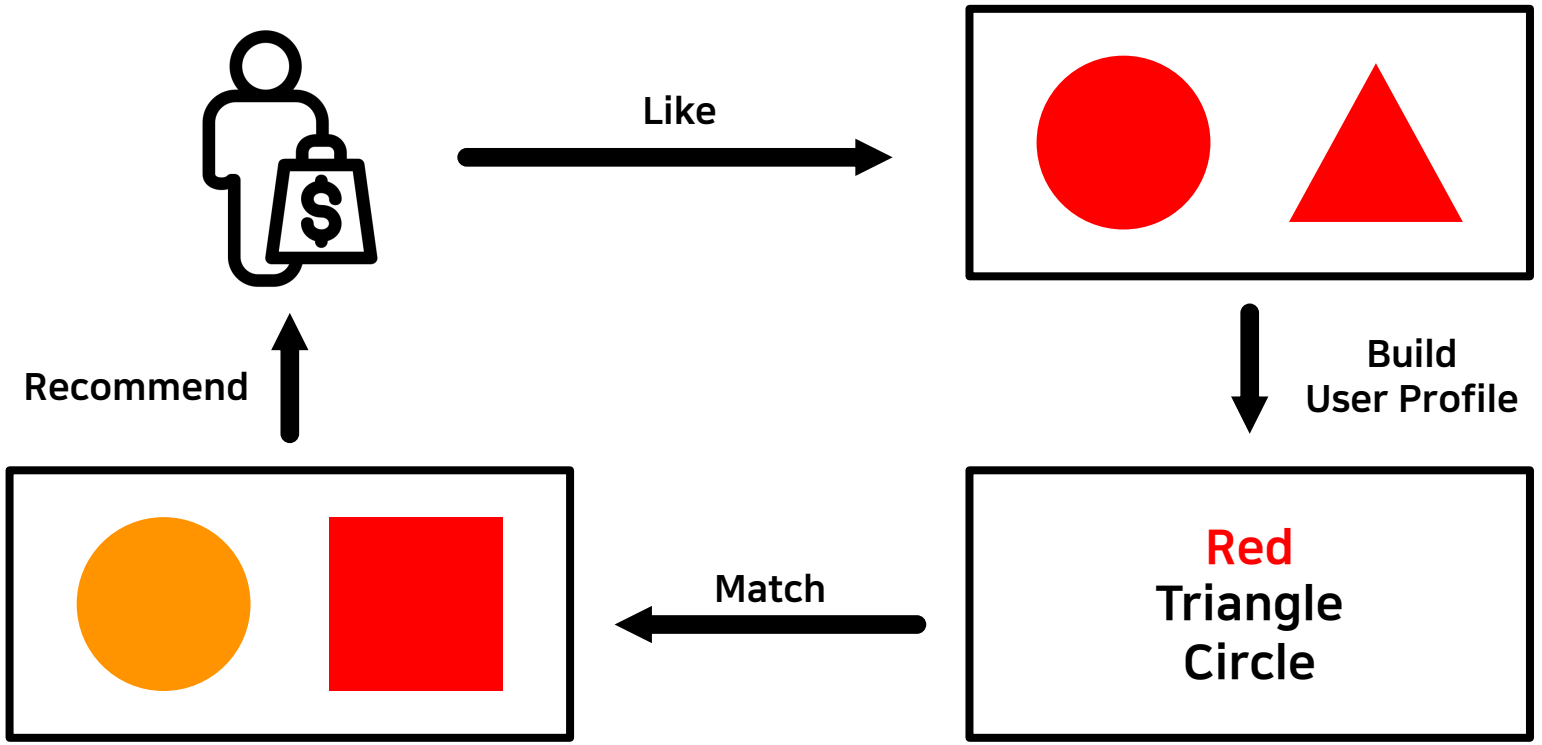
Content-based Recommendation

유저가 선호하는 아이템을 기반으로 해당 아이템과 유사한 아이템을 추천



Content-based Recommendation

유저가 선호하는 아이템을 기반으로 해당 아이템과 유사한 아이템을 추천



Item Profile

추천 대상이 되는 아이템에 대해서 Profile을 만들어야 함

Profile은 아이템이 가지고 있는 feature들로 구성됨

- **영화:** 작가, 제목, 배우, 장르, 감독
- **이미지, 동영상:** 메타데이터, 태그, 업로드한 사람
- **SNS:** 친구, 팔로잉/팔로워

아이템이 가진 다양한 속성(feature)를 어떻게 표현하면 가장 편할까? ➔ Vector의 형태

- 하나의 feature가 1개 혹은 1개 이상의 vector dimension에 표현됨
- Vector는 0/1 혹은 실수값으로 구성됨

Text Feature

문서(document)의 경우,

Item Profile: 중요한 단어들의 집합으로 표현될 수 있다.

그렇다면 중요한 단어를 어떻게 선정해야 하는가?

단어에 대한 중요도를 나타내는 스코어가 필요하다.

Text Mining에서 가장 많이 쓰는 기본적인 방법은 TF-IDF

문서 d에 등장하는 단어 w에 대해서,

단어 w가 문서 d에 많이 등장하면서, (Term Frequency)

단어 w가 전체 문서(D)에서는 적게 등장하는 단어라면, (Inverse Document Frequency)

➔ 단어 w는 문서 d를 설명하는 중요한 피쳐, TF-IDF 값이 높음!

TF - IDF

TF: 단어 w가 문서 d에 등장하는 횟수

$$TF(w,d) = freq_{w,d}$$

$TF(w,d) = \frac{freq_{w,d}}{\max_k(freq_{k,d})}$ (normalize TF to discount 'longer' document)

IDF: 전체 문서 가운데 단어 w가 등장한 비율의 역수

$IDF(w) = \log\frac{N}{n_w}$ (N: 전체 문서 개수, n_w: w가 등장한 문서 개수)

IDF 값의 변화가 크기 때문에 smoothing을 위해 logarithm을 사용함.

$$TFIDF(w,d) = TF(w,d) \cdot IDF(w)$$

TF - IDF 예시

	w1	w2	w3	w4	w5	w6
d1	2	2	1	3	0	2
d2	0	0	0	1	0	3
d3	0	0	1	1	0	2
d4	2	0	2	1	2	3

TF - IDF 예시

	w1	w2	w3	w4	w5	w6
d1	2	2	1	3	0	2
d2	0	0	0	1	0	3
d3	0	0	1	1	0	2
d4	2	0	2	1	2	3
IDF	log(4/2)	log(4/1)	log(4/3)	log(4/4)	log(4/1)	log(4/4)

TF - IDF 예시

	w1	w2	w3	w4	w5	w6
d1	2	2	1	3	0	2
d2	0	0	0	1	0	3
d3	0	0	1	1	0	2
d4	2	0	2	1	2	3
IDF	log(4/2)	log(4/1)	log(4/3)	log(4/4)	log(4/1)	log(4/4)

$$TFIDF(w2, d1) = 2 \cdot \log\frac{4}{1} = 1.2$$

$$TFIDF(w3, d1) = 1 \cdot \log\frac{4}{3} = 0.12$$

$$TFIDF(w4, d1) = 3 \cdot \log\frac{4}{4} = 0.0$$

TF - IDF 예시

	w1	w2	w3	w4	w5	w6
d1	2•log(4/2)	2•log(4/1)	1•log(4/3)	3•log(4/4)	0•log(4/1)	2•log(4/4)
d2	0•log(4/2)	0•log(4/1)	0•log(4/3)	1•log(4/4)	0•log(4/1)	3•log(4/4)
d3	0•log(4/2)	0•log(4/1)	1•log(4/3)	1•log(4/4)	0•log(4/1)	2•log(4/4)
d4	2•log(4/2)	0•log(4/1)	2•log(4/3)	1•log(4/4)	2•log(4/1)	3•log(4/4)
IDF	log(4/2)	log(4/1)	log(4/3)	log(4/4)	log(4/1)	log(4/4)

문서 내 등장하는 단어의 개수가 총 6개인 경우, 문서를 표현하는 item profile vector는 6차원 vector가 된다.
(상위 표의 개별 row)

User Profile

Item Profile을 모두 구축했으나, 우리가 해야 할 일은 유저에게 아이템을 추천하는 것이다.

➔ User Profile 구축이 필요함

User Profile

Item Profile을 모두 구축했으나, 우리가 해야 할 일은 유저에게 아이템을 추천하는 것이다.

➔ User Profile 구축이 필요함

User Profile

- 유저가 과거에 선호했던 Item List가 있고 개별 Item은 TF-IDF로 Vectorize됨
- 유저에 매핑된 Item들의 Vector들이 결국 User Profile이 된다.
- Simple: 선호한 Item Vector들의 average 사용
- Variant: 유저가 아이템에 내린 선호도로 Normalize한 average 사용

User Profile Vector

유저 u에 대해서 선호하는 아이템 $I = \{i_1, \dots, i_n\}$가 존재하고 해당 아이템에 대한 선호도는 $r_{u,i}$

아이템 i에 대하여 TF-IDF를 통해 얻어진 벡터를 V_i 일 때,

Simple

$$\mathrm{U} = \frac{\sum_{i=1}^{n} \mathrm{V}_i}{n}$$

Variant

$$\mathrm{U} = \frac{\sum_{i=1}^{n} r_{u,i}\mathrm{V}_i}{\sum_{i=1}^{n} r_{u,i}}$$

U는 아이템 Vector와 동일한 차원을 갖는다.

➔ 그렇다면 User Vector를 가지고 어떻게 추천을 하는가?

User Profile Vector 예시

유저가 d1, d3을 선호했다면, 해당 유저의 프로파일 벡터는 다음과 같다.

	w1	w2	w3	w4	w5	w6
d1	2•log(4/2)	2•log(4/1)	1•log(4/3)	3•log(4/4)	0•log(4/1)	2•log(4/4)
d2	0•log(4/2)	0•log(4/1)	0•log(4/3)	1•log(4/4)	0•log(4/1)	3•log(4/4)
d3	0•log(4/2)	0•log(4/1)	1•log(4/3)	1•log(4/4)	0•log(4/1)	2•log(4/4)
d4	2•log(4/2)	0•log(4/1)	2•log(4/3)	1•log(4/4)	2•log(4/1)	3•log(4/4)
IDF	log(4/2)	log(4/1)	log(4/3)	log(4/4)	log(4/1)	log(4/4)

$$User\ Vector = \frac{v_{d1} + v_{d3}}{2}$$

Cosine Similarity

주어진 두 벡터 X, Y에 대하여,

$$\cos(\theta) = \cos(\mathrm{X}, \mathrm{Y}) = \frac{\mathrm{X} \cdot \mathrm{Y}}{|\mathrm{X}||\mathrm{Y}|} = \frac{\sum_{i=1}^{N} X_i Y_i}{\sqrt{\sum_{i=1}^{N} X_i^2} \sqrt{\sum_{i=1}^{N} Y_i^2}}$$

두 벡터의 각도를 이용하여 구할 수 있는 유사도

직관적으로 두 벡터가 가리키는 방향이 얼마나 유사한 지를 의미함

- 두 벡터의 방향이 비슷할수록 1에 가까움
- 방향이 정반대인 경우 -1에 가까움

두 벡터의 차원은 같아야 함

유저에게 추천을 해보자

User Vector u, Item Vector i 에 대해서,

$$score(u,i) = \cos(\mathrm{u},\mathrm{i}) = \frac{\mathrm{u}\cdot\mathrm{i}}{|\mathrm{u}|\cdot|\mathrm{i}|}$$

User Vector는 Item Vector의 aggregation된 형태로 만들어짐

둘의 유사도가 클수록 해당 아이템이 유저에게 관련성이 높다

➔ 만약 유저 u가 i에 대해 가질 선호도를 정확하게 예측하고 싶다면?

정확한 평점 예측을 해보자

User Vector를 따로 구하지 않고, 유저가 선호도를 표시한 모든 Item Vector를 활용한다.

유저 u가 선호하는 아이템 $I = \{i_1, \dots, i_N\}$ 의 Item vector는 $V = \{\mathrm{v}_1, \dots, \mathrm{v}_n\}$, 평점은 $r_{ui}\ for\ i \in I$ 일 때, 새로운 아이템 i' 에 대해서 평점을 예측해보자.

i'와 I에 속한 아이템 i 의 유사도

$$sim(i, i') = \cos(\mathrm{v}_i, \mathrm{v}_{i'})$$

$sim(i', i)\ for\ i \in I$ 를 weight로 사용하여 i'이 평점을 추론한다.

$$prediction(i') = \frac{\sum_{i=1}^{N} sim(i, i') \cdot r_{u,i}}{\sum_{i=1}^{N} sim(i, i')}$$

TF - IDF 활용 평점 예측

특정 유저에 대해서 선호한 영화가 3개 있을 때,

m1: r_{m1}=3.0, $v_{m1} = [0.2, 0.4, 1.2, 1.5]$

m2: r_{m2}=2.5, $v_{m2} = [0.4, 0.7, 0.3, 0.5]$

m3: r_{m3}=4.0, $v_{m3} = [0.3, 1.2, 1.0, 1.0]$

예측 하려는 영화 m4의 $v_{m4} = [0.4, 1.4, 3.1, 1.0]$ 이라면,

$\text{sim}(m4, m1) = \cos(v_{m4}, v_{m1}) = 0.83$

$\text{sim}(m4, m2) = \cos(v_{m4}, v_{m2}) = 0.72$

$\text{sim}(m4, m3) = \cos(v_{m4}, v_{m3}) = 0.88$

$$prediction(m4) = \frac{0.83 \cdot 3.0 + 0.72 \cdot 2.5 + 0.88 \cdot 4.0}{0.83 + 0.72 + 0.88} = 3.2$$

컨텐츠 기반 추천의 장단점

장점

유저에게 추천을 할 때 다른 유저의 데이터가 필요하지 않음

새로운 아이템 혹은 인기도가 낮은 아이템을 추천할 수 있음

추천 아이템에 대한 설명(Explanation)이 가능함

단점

아이템의 적합한 피처를 찾는 것이 어려움

한 분야/장르의 추천 결과만 계속 나올 수 있음 (Overspecialization)

다른 유저의 데이터를 활용할 수 없음

TF-IDF 및 컨텐츠 기반 추천 모델 실습

2. 협업 필터링의 원리

Collaborative Filtering (협업 필터링)

정의

'많은 유저들로부터 얻은 기호 정보'를 이용해 유저의 관심사를 자동으로 예측하게 하는 방법

Collaborative: 집단지성, 다수의 의견을 반영한다

많은 유저들의 데이터가 축적될수록 집단 지성이 높아지고, 추천은 정확해진다

예시

이 상품을 구매한 유저가 구매한 다른 상품들

이 영화를 선호하는 유저가 관람한 다른 영화들

Collaborative Filtering 예시

다른 고객이 함께 구매한 상품

삼성전자 갤럭시북 플렉스2 미스틱 블랙 노트북…

1,927,000원 로켓배송

★★★★★ (17)

LG전자 10세대 코어i7 윈10탑재 17형 LG 그램 2020년형…

1,679,000원

★★★★½ (111)

올뉴프레임 노트북 파우치, 달빛숲

19,250원 로켓배송

★★★★½ (92)

슈와츠코리아 노트북받침대 B, 블랙

22,700원

★★★★½ (136)

스코코 갤럭시북 플렉스2 15인치 NT950QDA 키보드 키스…

17,000원

★★★★★ (4)

알럽스킨 갤럭시북 플렉인치 종이질감 액정+외

26,900원

★★★★★ (1)

다른 고객이 함께 본 상품

2/3

마이크로소프트 서피스 랩탑 고 플래티넘 노트북 (i5-1035G1 31.62cm WIN10 Home), 윈도우 포함, 64GB, 4GB

748,000원 로켓배송

★★★★★ (1)

삼성전자 갤럭시북 S NT767XCM-K58 Earth Gold (Wi-Fi전용 i5-L16G7 33.7cm Win10 Home), 포함, eUFS 256GB, 8GB

1,109,000원 로켓배송

★★★★★ (40)

삼성전자 갤럭시북 이온 노트북 아우라실버 NT950XCR-A58A (i5-10210U), 미포함, NVMe 256GB, 16GB, WIN미포함, RAM 8GB + RAM 8GB + NVMe 256GB

1,499,000원 로켓배송

★★★★½ (230)

삼성전자 Plus 2 퓨어화이트 노트북 NT550XDZ-AD1AW (셀러론 6305 39.6cm), 포함, 256GB, 4GB

647,900원 로켓배송

★★★★★ (92)

LG전자 2021 그램17 옵시디안블랙 노트북 17ZD90P-GX5BK (i5-1135G7 43.1cm), 미포함, 256GB, 8GB

1,769,000원 로켓배송

★★★★½ (11)

Collaborative Filtering

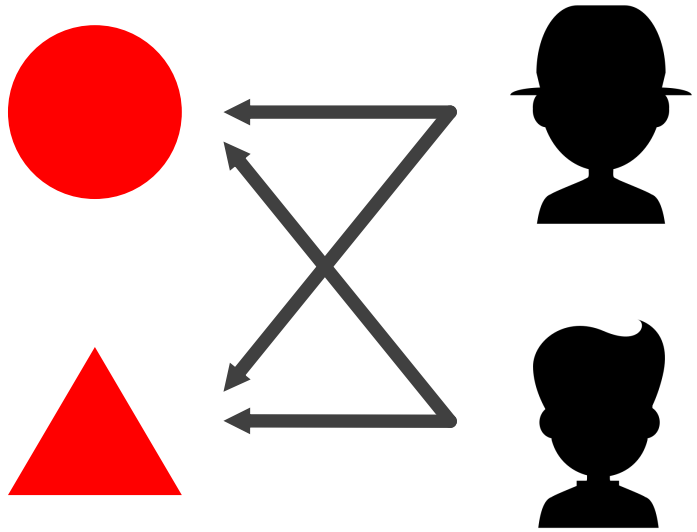
유저 A와 비슷한 성향을 갖는 유저들이 선호하는 아이템을 추천

아이템이 가진 속성을 사용하지 않으면서도 높은 성능을 보임

Collaborative Filtering

유저 A와 비슷한 성향을 갖는 유저들이 선호하는 아이템을 추천

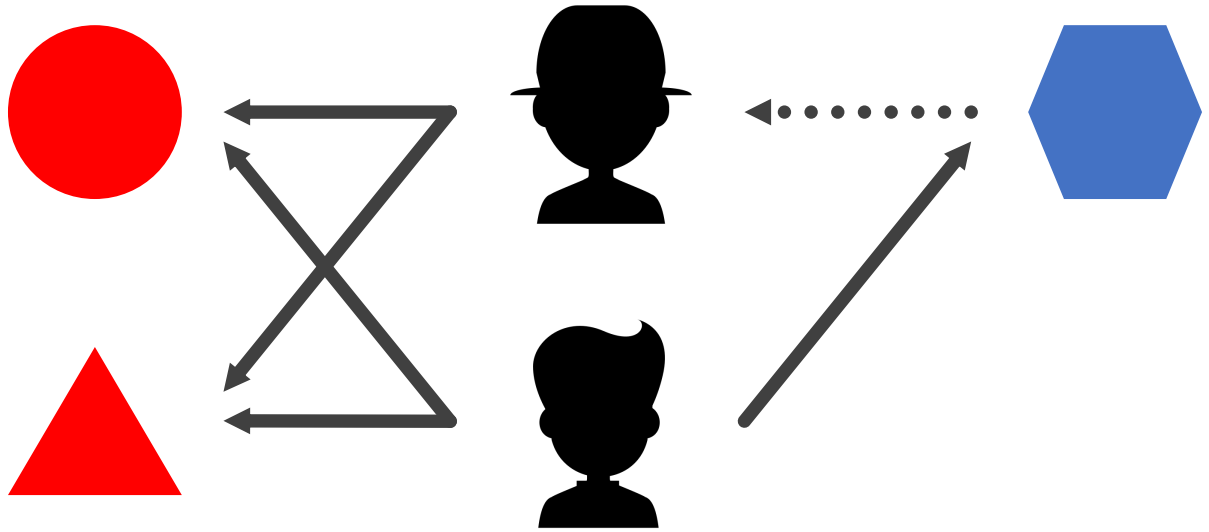
아이템이 가진 속성을 사용하지 않으면서도 높은 성능을 보임



Collaborative Filtering

유저 A와 비슷한 성향을 갖는 유저들이 선호하는 아이템을 추천

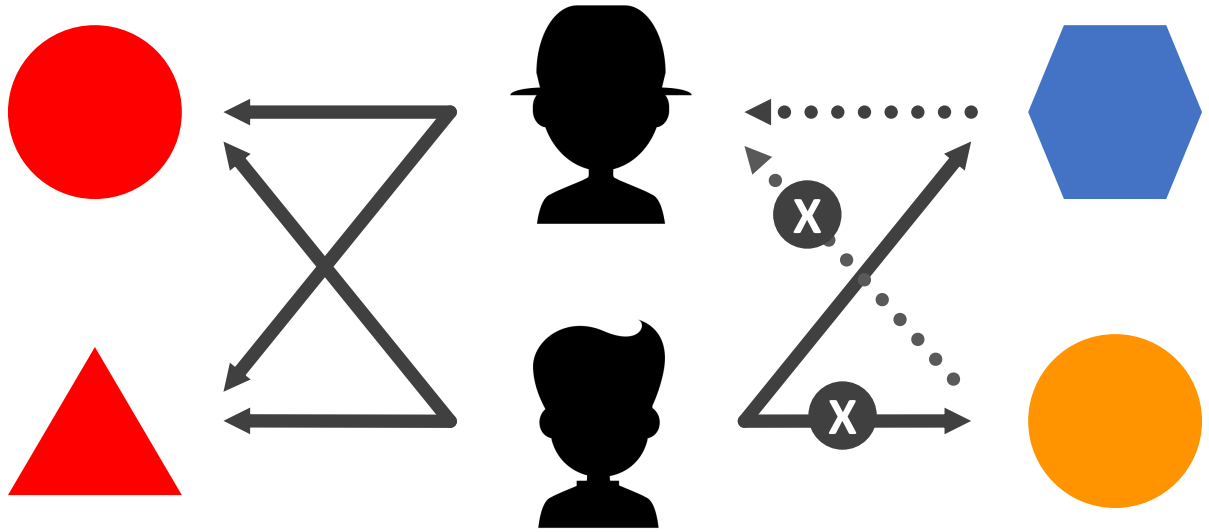
아이템이 가진 속성을 사용하지 않으면서도 높은 성능을 보임



Collaborative Filtering

유저 A와 비슷한 성향을 갖는 유저들이 선호하는 아이템을 추천

아이템이 가진 속성을 사용하지 않으면서도 높은 성능을 보임



Collaborative Filtering의 분류

Neighborhood-based Collaborative Filtering

- User-based
- Item-based

Model-based Collaborative Filtering

- Non-parametric (KNN, SVD)
- Matrix Factorization
- Deep Learning

Hybrid Collaborative Filtering

- Content-based Recommendation과의 결합

Collaborative Filtering의 목적

우리가 수행할 것은 결국 무엇인가?

➔ 유저 u의 아이템 i에 대한 평점을 예측하는 것

Collaborative Filtering의 목적

우리가 수행할 것은 결국 무엇인가?

➔ 유저 u의 아이템 i에 대한 평점을 예측하는 것

방법

주어진 데이터를 활용해 유저-아이템 행렬을 생성한다.

유사도 기준을 정하고, 유저 혹은 아이템 간의 유사도를 구한다.

주어진 평점과 유사도를 활용하여 행렬의 비어 있는 값(평점)을 예측한다.

Collaborative Filtering의 목적

우리가 수행할 것은 결국 무엇인가?

➔ 유저 u의 아이템 i에 대한 평점을 예측하는 것

방법

주어진 데이터를 활용해 유저-아이템 행렬을 생성한다.

유사도 기준을 정하고, 유저 혹은 아이템 간의 유사도를 구한다.

주어진 평점과 유사도를 활용하여 행렬의 비어 있는 값(평점)을 예측한다.

특징

구현이 간단하고 이해가 쉽다.

아이템이나 유저가 계속 늘어날 경우 확장성이 떨어짐 (Scalability)

Sparse한 데이터의 경우 성능이 저하됨 (Sparsity)

Sparsity

주어진 데이터를 활용해 유저 - 아이템 행렬을 만들어보자.

➔ 행렬의 대부분의 entry는 비어있다. (sparse matrix)

Sparsity

주어진 데이터를 활용해 유저 - 아이템 행렬을 만들어보자.

➔ 행렬의 대부분의 entry는 비어있다. (sparse matrix)

ex) Netflix

유저: 100m, 영화: 500k 존재

평균적으로 한 명의 유저가 몇 개의 영화를 봤을까?

Sparsity

주어진 데이터를 활용해 유저 - 아이템 행렬을 만들어보자.

➔ 행렬의 대부분의 entry는 비어있다. (sparse matrix)

ex) Netflix

유저: 100m, 영화: 500k 존재

평균적으로 한 명의 유저가 몇 개의 영화를 봤을까?

Collaborative Filtering을 적용하려면

적어도 sparsity가 99.5%를 넘지 않도록 하는 것이 좋음

sparsity: 행렬의 전체 entry 가운데 비어있는 비율

3. User-based, Item-based

User-based Collaborative Filtering

User-based

두 유저가 얼마나 유사한 아이템을 선호하는가?

유저간의 유사도를 구한 뒤, 나와 유사도가 높은 유저들이 선호하는 아이템을 추천한다.

User-based Collaborative Filtering

User-based

두 유저가 얼마나 유사한 아이템을 선호하는가?

유저간의 유사도를 구한 뒤, 나와 유사도가 높은 유저들이 선호하는 아이템을 추천한다.

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

User-based Collaborative Filtering

직관적으로 유저 B는 A와 비슷한 취향을 가졌기 때문에,

유저 B의 스타워즈에 대한 선호도는 높을 것으로 예측된다.

➔ 유저 A와 B의 유사도가 높다. (수학적으로 highly correlated)

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Rating Prediction

유저 B의 스타워즈에 대한 Rating은 어떻게 예측해야 할까?

Average

다른 유저들의 스타워즈에 대한 rating을 모두 사용하여 평균을 냄 (User A, C, D)

즉, User B의 입장에서 볼 때 A와 C의 rating을 동일하게 반영한다.

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Rating Prediction

유저 B의 스타워즈에 대한 Rating은 어떻게 예측해야 할까?

Average

다른 유저들의 스타워즈에 대한 rating을 모두 사용하여 평균을 냄 (User A, C, D)

즉, User B의 입장에서 볼 때 A와 C의 rating을 동일하게 반영한다.

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	$\frac{5+1+3}{3}$	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Rating Prediction

유저 B의 스타워즈에 대한 Rating은 어떻게 예측해야 할까?

Weighted Average

유저 간의 유사도 값을 weight로 사용하여 rating의 평균을 냄

User B의 입장에서 볼 때, User A의 rating은 많이 반영되고 User C의 rating은 적게 반영되어 예측 평점이 구해짐

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Rating Prediction

유저 B의 스타워즈에 대한 Rating은 어떻게 예측해야 할까?

Weighted Average

유저 간의 유사도 값을 weight로 사용하여 rating의 평균을 냄

User B의 입장에서 볼 때, User A의 rating은 많이 반영되고 User C의 rating은 적게 반영되어 예측 평점이 구해짐

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	$\frac{\sum_u sim(B,u) \cdot r_{u,i_3}}{\sum_u sim(B,u)}$	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

수식으로 정리하면

유저 $u \in U$, 아이템 $i \in I$ 에 대해 평점 데이터 $r(u,i)$ 가 존재할 때, 유저 u의 아이템 i에 대한 평점을 예측해보자

아이템 i 에 대한 평점이 있으면서 유저 u 와 유사한 유저들의 집합을 Ω_i라고 하면,

1) average

$$\hat{r}(u,i) = \frac{\sum_{u' \in \Omega_i} r(u',i)}{|\Omega_i|}$$

2) weighted average

$$\hat{r}(u,i) = \frac{\sum_{u' \in \Omega_i} sim(u,u')r(u',i)}{\sum_{u' \in \Omega_i} sim(u,u')}$$

Absolute Rating의 문제점?

내가 평점을 내리는 기준은 다른 유저와 다르다.

어떤 유저는 전체적으로 높게 평점을 줄 수도 있고 반대로 낮게 줄 수도 있다.

긍정적 유저: 대부분 5점을 주고 부정적인 평가로 3점을 줌

부정적 유저: 대부분 1~2점을 주고 가끔 4점을 줌

$$\hat{r}(u,i) = \frac{\sum_{u' \in \Omega_i} r(u',i)}{|\Omega_i|}$$

Deviation을 사용하자

유저가 아이템에 내린 절대 평점을 사용하지 않는다.

대신 유저의 평균 평점에서 얼마나 높은지 혹은 낮은지, 그 편차를 사용한다.

어떤 유저의 평균이 2.5점인데, 5점을 줬다면 아주 높게 평가한것이다.

모든 아이템의 평점을 5점으로 준 유저는 아이템끼리의 비교가 어렵다.

$$dev(u,i) = r(u,i) - \overline{r_u}, for\ known\ rating$$

Deviation을 사용하자

모든 평점 데이터를 deviation 데이터로 바꾼 뒤, predicted rating이 아닌 predicted deviation을 구한다.

predicted rating = 유저 평균 rating + predicted deviation

$$dev(u,i) = r(u,i) - \overline{r_u}, \qquad for\ known\ rating$$

$$\widehat{dev}(u,i) = \frac{\sum_{u' \in \Omega_i} dev(u',i)}{|\Omega_i|} = \frac{\sum_{u' \in \Omega_i} r(u',i) - \overline{r_{u'}}}{|\Omega_i|},$$

$$\hat{r}(u,i) = \overline{r_u} + \frac{\sum_{u' \in \Omega_i} r(u',i) - \overline{r_{u'}}}{|\Omega_i|} = \overline{r_u} + \widehat{dev}(u,i)$$

Weighted Average with deviation

deviation과 유사도 기반 weighted average prediction을 결합하여 최종 수식을 구해보면 다음과 같다.

Using Deviation

$$\hat{r}(u,i) = \overline{r_u} + \frac{\sum_{u' \in \Omega_i} sim(u,u')\{r(u',i) - \overline{r_{u'}}\}}{\sum_{u' \in \Omega_i} sim(u,u')}$$

Using Absolute Rating (비교)

$$\hat{r}(u,i) = \frac{\sum_{u' \in \Omega_i} sim(u,u')r(u',i)}{\sum_{u' \in \Omega_i} sim(u,u')}$$

K Nearest Neighbors Collaborative Filtering

아이템 i 에 대한 평점 예측을 하기 위해서는, 아이템 i에 대해 평가를 한 유저(Ω_i)의 데이터를 사용해야 한다

Ω_i에 속한 모든 유저와의 유사도를 구해야 함

모든 유저를 사용할 경우 연산은 많아지고 오히려 성능이 떨어지기도 함

➔ Ω_i에 속한 유저 가운데 유저 u와 가장 유사한 K명의 유저를 이용해 평점을 예측한다 (KNN)

유사하다는 것은 우리가 정의한 유사도 값이 크다는 것을 의미함

보통 K = 25 ~ 50을 많이 사용하지만 직접 튜닝해야 하는 하이퍼 파라미터

Item-based Collaborative Filtering

Item-based

두 아이템이 유저로부터 얼마나 유사한 평가를 받았는가?

아이템 선호도를 바탕으로 연관성이 높은 다른 아이템을 추천, 아이템 간의 유사도를 구한다.

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Item-based Collaborative Filtering

직관적으로 스타워즈는 아이언맨, 헐크와의 유사도가 높다. 반대로 비포 선라이즈, 노팅힐은 스타어즈와의 유사도가 낮다.

➔ 따라서 유저 B의 스타워즈에 대한 평점은 아이언맨, 헐크와 비슷하게 높을 것이다.

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

Item-based Collaborative Filtering

스타워즈와 가장 유사도가 큰 아이언맨(0.7), 헐크(0.9)를 활용해 예측하면,

유저 B의 스타워즈에 대한 예측 평점은

$$\frac{0.7 \cdot 4 + 0.9 \cdot 5}{0.7 + 0.9} = 4.6$$

	아이언맨	헐크	스타워즈	비포선라이즈	노팅힐
User A	5	4.5	5	2	1
User B	4	5	?	1	2
User C	2	1	1	4	5
User D	3	3	3	3	3

User-based와 동일하게 Rating Prediction

유저 $u \in U$, 아이템 $i \in I$ 에 대해 평점 데이터 $r(u,i)$ 가 존재할 때, 유저 u 의 아이템 i 에 대한 평점을 예측해보자.

유저 u 가 평가를 한 다른 아이템 중에서 아이템 i와 유사한 아이템들의 집합을 Φ_u 라고 하면,

1) Average

$$\hat{r}(u,i) = \frac{\sum_{i' \in \Phi_u} r(u,i')}{|\Phi_u|}$$

2) Weighted Average

$$\hat{r}(u,i) = \frac{\sum_{i' \in \Phi_u} sim(i,i') r(u,i')}{\sum_{i' \in \Phi_u} sim(i,i')}$$

User-based와 동일하게 Rating Prediction

Using Deviation

1) Average

$$\hat{r}(u,i) = \overline{r_i} + \frac{\sum_{i' \in \Phi_u} r(u,i') - \overline{r_{i'}}}{|\Phi_u|}$$

2) Weighted Average

$$\hat{r}(u,i) = \overline{r_i} + \frac{\sum_{i' \in \Phi_u} sim(i,i')\{r(u,i') - \overline{r_{i'}}\}}{\sum_{i' \in \Phi_u} sim(i,i')}$$

User-based vs Item-based

User-based

- 구현이 쉽고 유사한 Neighborhood의 수 K가 늘어날 수록 성능이 높아짐
- Item-based보다 더 다양한 추천 결과들이 제공됨 (Diversity)
- Sparsity, Cold Start에 좀 더 취약함
- Pearson 유사도를 사용할 때 성능이 높음

Item-based

- 보통 실제 서비스에서 User-based CF보다 높은 성능을 냄
- 아이템 간의 유사도를 사용하는 것이 더 Robust함
  - 아이템 기준의 Neighborhood들이 사용자 기준의 Neighborhood보다 훨씬 덜 변함
- 추천에 대한 이유를 설명하기 훨씬 쉬움
  - 유저가 과거에 선호했던 다른 아이템과 비슷하기 때문에 추천
- Cosine 유사도를 사용할 때 성능이 높음

Collaborative Filtering의 한계

1. Cold Start 문제

데이터가 충분하지 않다면 추천 성능이 떨어진다.

데이터가 전혀 없는 신규 유저, 아이템의 경우 추천이 불가능하다.

2. 계산 효율

유저와 아이템이 늘어날수록 유사도 계산이 늘어난다.

유저, 아이템이 많아야 정확한 예측을 하지만 반대로 시간이 오래걸린다.

3. Long-tail 추천의 한계

많은 유저들이 선호하는 소수의 아이템이 보통 CF 추천 결과로 나타남.

롱테일을 이루는 비주류의 아이템이 추천되기 어려움.

4. 유사도 개념 이해하기

Cosine Similarity

주어진 두 벡터 X, Y에 대하여,

$$\cos(\theta) = \cos(\mathrm{X}, \mathrm{Y}) = \frac{\mathrm{X} \cdot \mathrm{Y}}{|\mathrm{X}||\mathrm{Y}|} = \frac{\sum_{i=1}^{N} X_i Y_i}{\sqrt{\sum_{i=1}^{N} X_i^2} \sqrt{\sum_{i=1}^{N} Y_i^2}}$$

두 벡터의 각도를 이용하여 구할 수 있는 유사도

직관적으로 두 벡터가 가리키는 방향이 얼마나 유사한 지를 의미함

- 두 벡터의 방향이 비슷할수록 1에 가까움
- 방향이 정반대인 경우 -1에 가까움

두 벡터의 차원은 같아야 함

Mean Squared Difference Similarity

주어진 유저-아이템 스코어에 대하여,

$$msd(u,v) = \frac{1}{|I_{uv}|} \cdot \sum_{i \in I_{uv}} (r_{ui} - r_{vi})^2 , \qquad msd\_sim(u,v) = \frac{1}{msd(u,v)+1}$$

$$msd(i,j) = \frac{1}{|U_{ij}|} \cdot \sum_{u \in U_{ij}} \left(r_{ui} - r_{uj}\right)^2 , \qquad msd\_sim(i,j) = \frac{1}{msd(i,j)+1}$$

추천 시스템에서 주로 사용되는 유사도

각 기준(유저, 아이템)에 대한 점수의 차이를 계산, 유사도는 유클리드 거리에 반비례

분모에 1을 더하는 이유는 분모가 0이 되는 것을 방지하는 일종의 smoothing

Jaccard Similarity

주어진 집합 A, B에 대하여,

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$

집합의 개념을 사용한 유사도

cosine, pearson과 달리 길이가 달라도 이론적으로 유사도를 구할 수 있음

두 집합이 얼마나 유사한 아이템를 공유하고 있는가를 나타냄

- 두 집합이 가진 아이템이 모두 같으면 1
- 두 집합에 겹치는 아이템이 하나도 없으면 0

Pearson Similarity (Pearson Correlation)

주어진 벡터 X, Y에 대해서

$$pearson\_sim(\mathrm{X}, \mathrm{Y}) = \frac{\sum_{i=1}^{N}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{N}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{N}(Y_i - \overline{Y})^2}}$$

각 벡터를 표본평균으로 정규화한 뒤에 cosine 유사도를 구한 값

직관적으로 해석하면 (X와 Y가 함께 변하는 정도) / (X와 Y가 따로 변하는 정도)

1에 가까우면 양의 상관관계, 0일 경우 서로 독립, -1에 가까울수록 음의 상관관계를 나타냄

유사도와 협업 필터링 실습