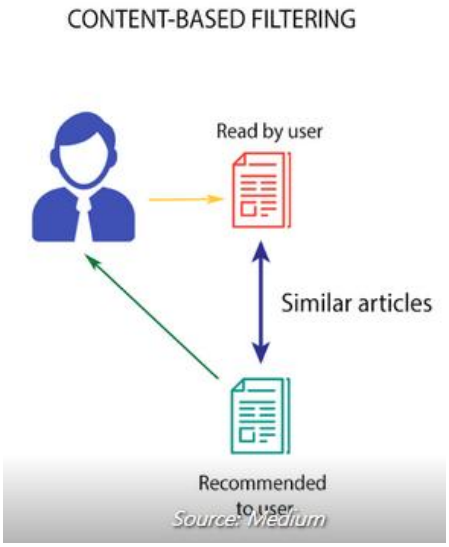
Comprehensive Guide to build a Recommendation Engine from scratch (in Python)

https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/

- Content Based filtering



사용자가 과거에 좋아했던 item과 비슷한 item을 추천해주는 알고리즘
가지고 있는 정보를 Profile Vector, Item Vector나타낸 뒤, 상품들 간의 유사도를 측정한다.

Cosine similarity.

$$sim(A,B) = \cos(\theta) = \frac{A \cdot B}{\|A\|\|B\|}$$

Euclidean Distance

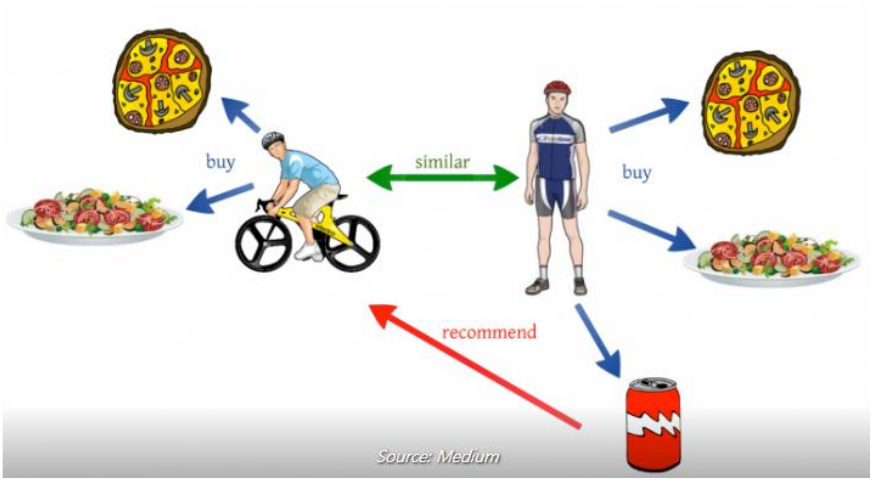
$$\text{Euclidean Distance} = \sqrt{(x_1 - y_1)^2 + \ldots + (x_N - y_N)^2}$$

Pearson`s Correlation

$$sim(u,v) = \frac{\sum(r_{ui} - \bar{r}_u)(r_{vi} - \bar{r}_v)}{\sqrt{\sum(r_{ui} - \bar{r}_u)^2}\sqrt{\sum(r_{vi} - \bar{r}_v)^2}}$$

과거, 구매하지 않고 호의를 표하지 않은 상품에 대해서는 추천을 하지 않는다는 문제가 있다.

- User – User Collaborative filtering



유저들간의 유사성점수를 찾는다. 서로 유사한 제품을 선택하는 유저집단에서. 어떤 유저가 구매한 물건을 다른 유저에게 추천을 한다.

The prediction *Pu,i* is given by:

$$P_{u,i} = \frac{\sum_v (r_{v,i} * s_{u,v})}{\sum_v s_{u,v}}$$

Here,

- *Pu,i* is the prediction of an item
- *Rv,i* is the rating given by a user *v* to a movie *i*
- *Su,v* is the similarity between users

1. User u와 v사이의 유사성을 예측하기 위해서 pearson correlation을 계산할 수 있다.
2. 유저와 평점들에 기반하여 평가된 아이템을 찾는다. 유저들간의 상관계수를 계산한다.
3. 예측들은 유사도를 이용해서 계산 될 것이다. 높은 correlation을 가진 유저들은 비슷한 경향이 있다.
4. 예측값에 기반하여 추천을 한다.

User/Movie	x1	x2	x3	x4	x5	Mean User Rating
A	4	1	–	4	–	3
B	–	4	–	2	3	3
C	–	1	–	4	4	3

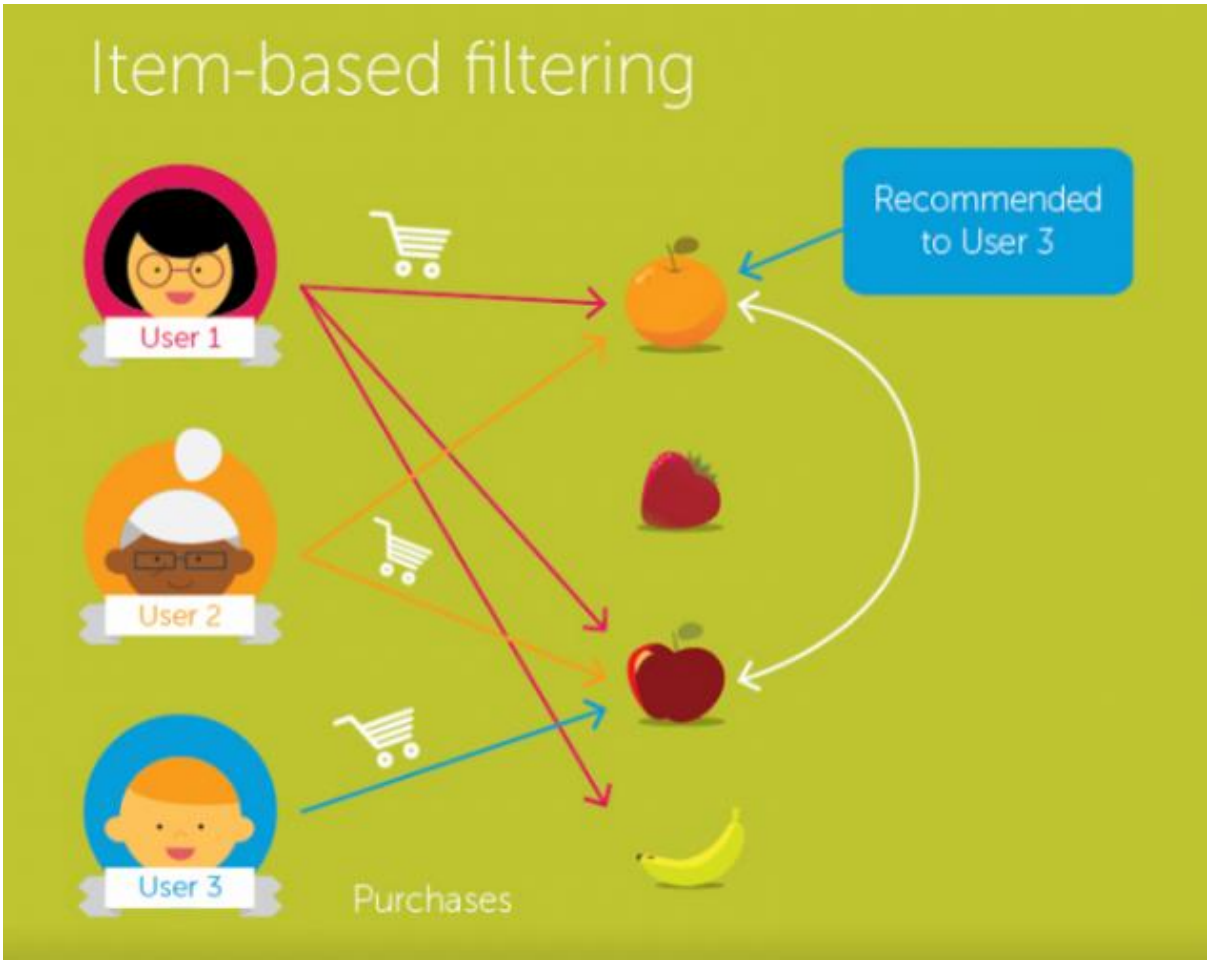
$r_{AC} = [(1-3)*(1-3) + (4-3)*(4-3)]/[((1-3)^2 + (4-3)^2)^{1/2} * ((1-3)^2 + (4-3)^2)^{1/2}] = 1$

$r_{BC} = [(4-3)*(1-3) + (2-3)*(4-3) + (3-3)*(4-3)]/[((4-3)^2 + (2-3)^2 + (3-3)^2)^{1/2} * ((1-3)^2+(4-3)^2 + (4-3)^2)^{1/2}] = -0.866$

AC의 correlation이 BC의 correlation보다 좋다. User A와 C는 비슷한 영화를 좋아할 가능성이 높다.

사용자들 사이에 유사성을 계산하는데 시간이 많이 걸리기에 알고리즘은 사용자의 수가 적을 때 유용하다. 반면 Item collaborative filtering은 item수가 많아질수록 효과적인 추천이 가능하다.

- Item – Item Collaborative filtering



Items 쌍들 간의 유사성을 계산한다. 과거에 사용자들이 좋아했던 비슷한 영화들을 추천한다.

$$P_{u,i} = \frac{\sum_N (s_{i,N} * R_{u,N})}{\sum_N (|s_{i,N}|)} \qquad sim(i,j) = cos(\vec{i}, \vec{j}) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 * \|\vec{j}\|_2}$$

User/Movie	x1	x2	x3	x4	x5
A	4	1	2	4	4
B	2	4	4	2	1
C	–	1	–	3	4
Mean Item Rating	3	2	3	3	3

영화에 대한 공통 사용자를 찾자. 영화X1,X4 – A,B 영화X1,X5 – A,B

$$C_{14} = [(4-3)*(4-3) + (2-3)*(2-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (2-3)^2)^{1/2}] = 1$$

$$C_{15} = [(4-3)*(4-3) + (2-3)*(1-3)] / [((4-3)^2 + (2-3)^2)^{1/2} * ((4-3)^2 + (1-3)^2)^{1/2}] = 0.94$$

영화X1, X4에 대한 유사성이 더 높다. 어떤 User가 영화x1를 검색하면 영화x4를 추천 할 것이다.

- Cold Start - 새로운user, 새로운item이 데이터 셋에 추가가 된다면 어떻게 할 것인가?

1. Visitor Cold Start (새로운user)
   User에 대한 history가 없으면 시스템은 상품을 추천하기 어렵다.
   이를 해결하기 위해 가장 인기 있는 상품을 추천한다.

2. Product Cold start (새로운item)
   User의 행동은 어떤 item의 가치를 결정하는데 중요하다. item들간의 상호작용이 많을수록 적절한 사용자에게 제품을 권장하기 쉽다. 하지만 새로운 item이 들어왔을 때는 어떻게 할 것 인가?
   content기반 filtering을 이용한다. 새로운 item에 대한 내용은 권장사항으로 사용하고 다음에는 user의 행동에 따라 조치를 취한다.