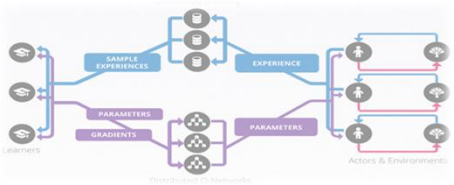



Audio Recommendation in the Sound Neural Network

AutoML 연구실
학부연구생 조성현
20.11.24

Introduction

Typical Filter Types in Recommendation

- **Collaborative Filtering**
  - **Memory-based**
    - User-based
    - Item-based
    - 코사인 유사도
    - 피어슨 유사도
  - **Model-based**
    - Matrix Factorization
    - Clustering
    - Bayesian Network
- **Content Based Filtering**
  - Term Frequency – Inverse Document Frequency
- **Hybrid Recommender Systems**
  - Demographic Filtering
  - Knowledge Based Filtering

Collaborative Filtering

- ## Memory-based Filtering

- ### User-based

두 사용자가 얼마나 유사한 항목을 선호했는지를 기준으로 한다. 이 때, **두 사용자 간의 유사도**는 두 벡터 간의 유사도로 정의할 수 있다.

두 벡터 간의 유사도를 구하기 위해서 다양한 방법이 사용될 수 있는데 대개는 **코사인 유사도, 피어슨 유사도**가 이용된다.

$$similarity = \cos\theta = \frac{A \cdot B}{\|A\|\|B\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$,where A_i and B_i are components of vector A and B respectively

<코사인 유사도>

코사인 유사도에서는 유저마다의 **개인적인 평가 성향을 반영하지 못한다**는 단점이 있다.

$$similarity = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_u})(R_{u,j} - \overline{R_u})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_u})^2}\sqrt{\sum_{u \in U}(R_{u,j} - \overline{R_u})^2}}$$

<피어슨 유사도>

- ### Item-based

아이템 기반도 앞선 사용자 기반과 매우 유사한 과정을 거치지만, 여기서는 **아이템들에 대한 유사도를 계산**한다.

Collaborative Filtering

<table>
<tr><th>항목</th><th>User-based</th><th>Item-based</th></tr>
<tr><td>개념</td><td>선호 성향이 비슷한 사용자들을 같은 그룹화,
동일 그룹 선호 상품 추천</td><td>과거 구매 아이템 기반 아이템과 선호 연관성 높은 다른 아이템 추천</td></tr>
<tr><td>설명</td><td>
<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th></tr>
<tr><td>User 1</td><td>O</td><td>O</td><td>O</td><td>O</td></tr>
<tr><td>User 2</td><td></td><td>O</td><td></td><td></td></tr>
<tr><td>User 3</td><td>?</td><td>O</td><td>O</td><td>?</td></tr>
</table>
User 1, 3 패턴 유사 → User 3에게 A, D 추천
</td><td>
<table>
<tr><th></th><th>A</th><th>B</th><th>C</th><th>D</th></tr>
<tr><td>User 1</td><td>O</td><td></td><td>O</td><td>O</td></tr>
<tr><td>User 2</td><td>O</td><td>O</td><td>O</td><td></td></tr>
<tr><td>User 3</td><td>O</td><td></td><td>?</td><td></td></tr>
</table>
Item A와 C 연관성 높음 → A 구매고객에게 C 추천
</td></tr>
</table>

Collaborative Filtering

- ## 코사인 유사도

두 특성 벡터 간의 유사 정도를 코사인으로 표현했다.

$$\boldsymbol{similarity} = \cos\boldsymbol{\theta} = \frac{\boldsymbol{A}\cdot\boldsymbol{B}}{\|\boldsymbol{A}\|\|\boldsymbol{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$$,where A_i and B_i are components of vector A and B respectively, $-1 \le \cos\boldsymbol{\theta} \le 1$

Ex) vector A(5, 5, 5, 5, 5), vector B(1, 1, 1, 1, 1)

$$\boldsymbol{similarity} = \frac{5*1+\ 5*1+\ 5*1+\ 5*1+\ 5*1}{\sqrt{25+25+25+25+25}*\sqrt{1+1+1+1+1}} = \frac{25}{25}$$

$= 1$

두 아이템에 대한 평가는 극명하게 갈림에도 불구하고 두 벡터의 유사도는 1이다.

코사인 유사도는 유사도를 구할 때, 벡터의 크기가 아니라 **벡터의 방향(패턴)에 초점**을 두기 때문이다.

유저마다의 **개인적인 평가 성향을 반영하지 못한다**는 단점이 있다.

Collaborative Filtering

- **피어슨 유사도**

특정 점수가 너무 낮거나 높을 경우 유사도에 영향을 크게 미치는 경우를 방지하기 위해 상관계수를 사용한다.

$$similarity = \rho_{A,B} = \frac{Cov(A,B)}{\sigma_A \sigma_B} = \frac{E(A-\mu_A)(B-\mu_B)}{\sqrt{E(A-\mu_A)^2}\sqrt{E(B-\mu_B)^2}} \rightarrow \frac{\sum(A-\bar{A})(B-\bar{B})}{\sqrt{\sum(A-\bar{A})^2}\sqrt{\sum(B-\bar{B})^2}} = cos(A-\bar{A}, B-\bar{B}), -1 \le \rho_{A,B} \le 1$$

모평균(μ_A, μ_B) 대신
표본평균($\bar{A}, \bar{B}$) 대입

	A	B	C	D	E	F	Avg.	$A-\bar{A}$	$B-\bar{B}$	$C-\bar{C}$	$D-\bar{D}$	$E-\bar{E}$	$F-\bar{F}$
User 1	4	3	4			1	3.00	1	0	1	0	0	-2
User 2	4	1	2	2		1	2.00	2	-1	0	0	0	-1
User 3	4			3	1		2.67	1.33	0	0	0.33	-1.67	0
User 4	1	4	2		4		2.75	-1.75	1.25	-0.75	0	1.25	0
User 5	1		3	1	3	2	2.00	-1	0	1	-1	1	0

sim(1,2)=0.67
sim(1,3)=0.25
sim(1,4)=-0.39
sim(1,5)=0

User 1과 가장 유사도가 높은 User 2와 3을 바탕으로 User 1이 Item D를 얼마나 좋아할지 예측이 가능해진다.
가중치를 고려해서 계산하면, User 2 = 2점 * 0.67, User 3 = 3점 * 0.25

$\frac{(2*0.67)+(3*0.25)}{0.67+0.25} = 2.27$, 그러므로 User 1에게 추천 시 2.27점을 줄 것으로 추정이 가능하다.

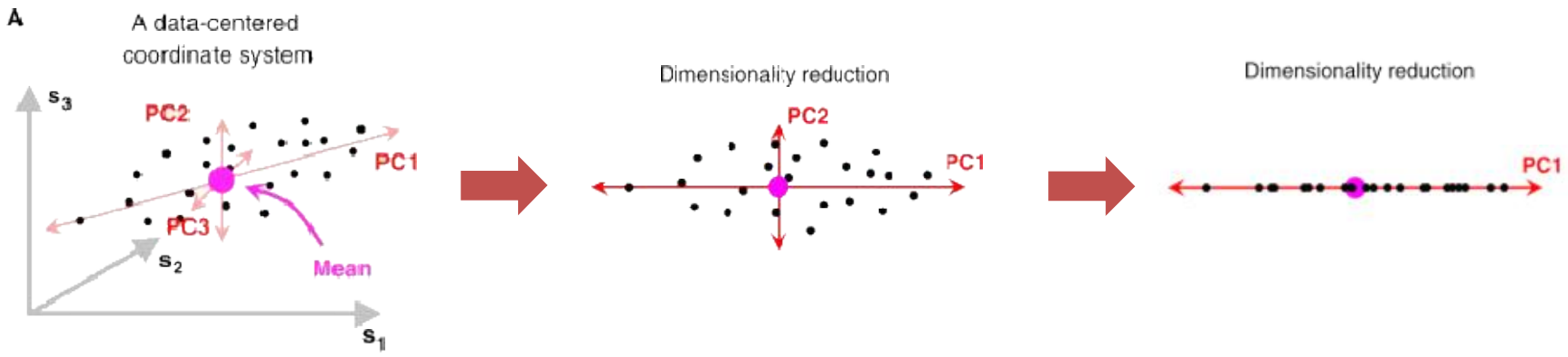
Collaborative Filtering

- **Model-based Filtering**

Data Mining, Machine Learning 알고리즘을 사용하여 User-Item matrix로부터 Model을 학습하여 **등급이 지정되지 않은 항목에 대한** 사용자의 **등급을 예측**한다.

Bayesian Networks, Clustering Models, Singular Value Decomposition 등 다양한 알고리즘이 존재한다.

Dimensionality Reduction Method를 통해 **Low-Dimensional Space**에서 **Robustness와 Accuracy를 보완한다**.



Collaborative Filtering

- **Matrix Factorization**

User-Item Matrix에서 **아직 평가되지 않은 빈 공간을 어떤 잠재요인(Latent Factor)이 있다고 가정**한다.

Item Rating Patterns로부터 학습하기 위해 두 개의 저차원 행렬의 곱으로 분해한다.(**Dimensional Reduction**)

분해된 행렬을 토대로 특정 User의 예상 선호도 값을 빈 공간에 넣어 추천한다.

- Dimensional Reduction의 장점

  쓸모 없는 Feature을 제거할 수 있다. → **복잡도 감소**

  해석과 시각화에 용이하다. → **이해도 증가**

  데이터를 처리하고 저장하기 쉽다. → **정확도 증가**

Collaborative Filtering

- **Eigenvalue Decomposition**

고유 벡터(Eigenvectors) – 어떤 벡터에 선형 변환을 취했을 때, 방향 변화 없이 크기만 변환되는 벡터

고유값(Eigenvalues) – 얼마나 확대 또는 축소했는지에 대한 값

N x N 정방행렬 A에 대하여 3개 행렬의 내적으로 나타낼 수 있다.

$$A = PDP^{-1} = (\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n}) \begin{bmatrix} \lambda_1 & 0 & 0 & 0 \\ 0 & \lambda_2 & 0 & 0 \\ 0 & 0 & \ddots & \vdots \\ 0 & 0 & \cdots & \lambda_n \end{bmatrix} (\boldsymbol{v_1}, \boldsymbol{v_2}, \cdots, \boldsymbol{v_n})^{-1}$$, λ_n:고유값, v_1 : λ_i에 해당하는 고유 벡터

정방행렬을 대각화함으로써 계산을 용이하게 만들어준다. Ex) $A^{100} = PD^{100}P^{-1}$

Collaborative Filtering

- **Singular Value Decomposition(SVD)**

User-Item의 평점 데이터를 행렬로 생각 하여 U 와 V 로 분리한다.

$$M = U\Sigma V^*$$

M 은 m*n 크기의 **직사각행렬**

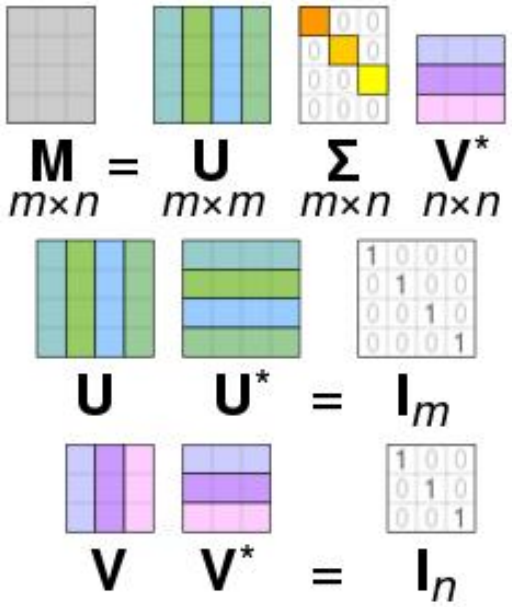
U는 MM^T를 고유값 분해해서 얻어진 m*m 크기의 직교행렬(차원+)

Σ는 m * n 크기의 주대각선 성분이 $\sqrt{\lambda_i}$인 대각행렬(길이 변화 유지를 위한$\sqrt{\ }$)

V^*는 V의 켤레전치 행렬로, n * n크기의 유니터리행렬(차원-)

V 는 $M^T M$를 고유값 분해해서 얻어진 n * n 크기의 직교행렬

유니터리 행렬 = 켤레전치가 역행렬과 같은 복소수 행렬



Collaborative Filtering

- **Principal Component Analysis(PCA)**

  데이터의 분포에 대한 최적의 Feature(주성분) 조합을 찾기 위한 방법이다.

  - Algorithm

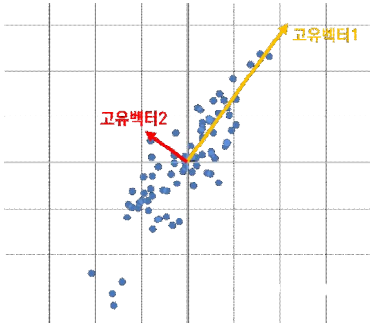
    (1) 데이터들의 **평균을 원점으로** 가정한다.

    (2) **분산이 최대인 축**을 찾는다. (분산이 최대인 축은 공분산 행렬의 고유 벡터이다.)

    (3) (2)축과 **직교하면서 분산이 최대인 축**을 찾는다.

    (4) 차원 수만큼의 축을 찾을 때까지 반복한다.

$$C(X)=\begin{bmatrix} Var(X_1) & Cov(X_1,X_2) & \cdots & Cov(X_1,X_K) \\ Cov(X_2,X_1) & Var(X_2) & \cdots & Cov(X_2,X_K) \\ \vdots & \vdots & \ddots & \vdots \\ Cov(X_K,X_1) & Cov(X_K,X_2) & \cdots & Var(X_k) \end{bmatrix}$$



Collaborative Filtering

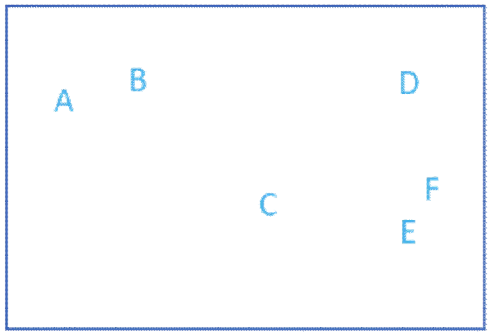
- **Clustering**

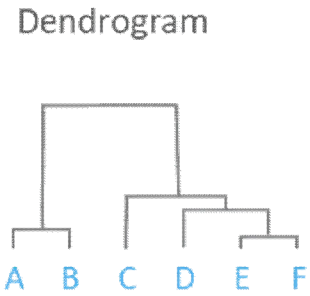
주어진 데이터들의 **특성을 고려해 데이터 집단을 정의**하고 **데이터 집단의 대표점을 찾는** Unsupervised Learning이다.

각 Cluster의 대표 값만 확인해 전체 데이터의 특성을 파악할 수 있다.

- **Hierarchical Clustering**

가장 가까운 Cluster끼리 반복적으로 결합한다.(가까움은 Centroid의 Euclidean Distance로 정의된다.)





Collaborative Filtering

- **K-means Clustering**

  각 구간을 나눈 다음 **Centroid를 찾고 Centroid를 기준으로 구간을 다시 나누고 변경 사항이 있을 경우 다시 Centroid를 찾아가는 방식**

- Algorithm

  데이터셋에서 **K개의 Centroids를 임의로 지정**한다. (K를 1씩 늘리면서 Centroid 간의 평균 거리가 더 이상 많이 감소하지 않는 경우의 K를 선택)

  각 데이터들을 **가장 가까운 Centroid가 속한 그룹에 할당**한다.

  할당된 결과를 바탕으로 **Centroids를 초기화**한다.

  Centroids가 크게 변하지 않을 때까지 **반복**한다.

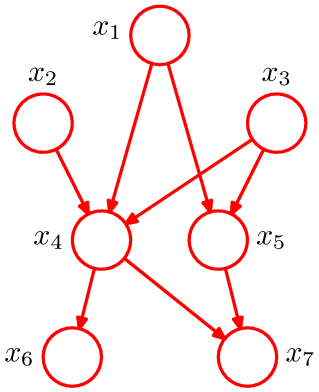
Collaborative Filtering

- **Bayesian Network**

복잡한 시스템의 모델링을 비순환그래프를 통해 조건부 독립으로 표현하는 **Probabilitistic Graphical Model**이다.

조건부 확률 P(A|B) – B가 참일 때, A가 참일 확률 , 결합 확률 P(A,B) – A와 B가 동시에 일어날 확률

P(A|B) = P(A,B) / P(B)



$P(X) = P(X_1)P(X_2)P(X_3)P(X_4 \mid X_1, X_2, X_3)P(X_5 \mid X_1, X_3)P(X_6 \mid X_4)P(X_7 \mid X_4\ X_5)$

일반화하면, $P(X) = \prod_{k=1}^{K} P(x_k|pa_k)$,pa_k는x_k의부모 노드의 집합이다.

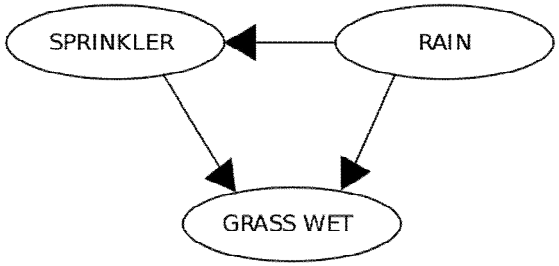
방향성 링크들이 Cycle을 형성하지 않은 Directed Acyclic Graph(DAG)이다.

Collaborative Filtering

- **Bayesian Network**

잔디가 젖었다면 비였을 확률은?

SPRINKLER		
RAIN	T	F
F	0.4	0.6
T	0.01	0.99



RAIN	
T	F
0.2	0.8

		GRASS WET	
SPRINKLER	RAIN	T	F
F	F	0.0	1.0
F	T	0.8	0.2
T	F	0.9	0.1
T	T	0.99	0.01

$$P(R=T|G=T) = \frac{p(G=T,R=T)}{P(G=T)}$$

$$= \frac{\sum_{S\in\{T,F\}} P(G=T,S,R=T)}{\sum_{S,R\in\{T,F\}} P(G=T,S.R)}$$

$$= \frac{(0.99*0.01*0.2=0.00198_{TTT})+(0.8*0.99*0.2=0.1584_{TFT})}{0.00198_{TTT}+0.288_{TTF}+0.1584_{TFT}+0_{TFF}}$$

$$\approx 35.77\%$$

Collaborative Filtering

- **Memory-based vs. Model-based**

Memory-based	Model-based
확장성 ↓	실제 Dataset보다 작은 Model로 인해 Dataset의 크기가 매우 크더라도 **효율적**으로 사용 가능
예측 속도 ↓	**예측 속도 ↑**
Overfitting ↑	**Overfitting ↓**
데이터 추가하기 쉬움(**Flexibility**)	시간과 리소스가 많이 소모되기 때문에 데이터를 추가하기 어려움(**Inflexibility**)
예측에 모든 데이터를 사용함(**Prediction Quality ↓**)	모든 데이터를 사용하지 않기 때문에 정확한 예측된 값 얻지 못할 수 있다.(**Prediction Quality ↑**)

Collaborative Filtering

- **장점**
  - 아이템 종류에 상관 없이 **구현이 쉽고 잘 동작**한다.
  - **특징을 선택할 필요가 없다.**

- **단점**
  - **Cold Start** - 시스템이 작동하기 위해서는 **User-Item Interactions에 대한 충분한 정보**가 있어야 한다.
  - **First rater** - 새로운 User/Item을 추가할 때 **기존의 Interactions가 없기 때문에 Prior 정보를 가지고 있지 않다.**
  - **Sparsity** – 빈 User/Ratings Matrix때문에 **동일한 아이템을 등급 매긴 사용자를 찾기 힘들다.**

Content-based Filtering

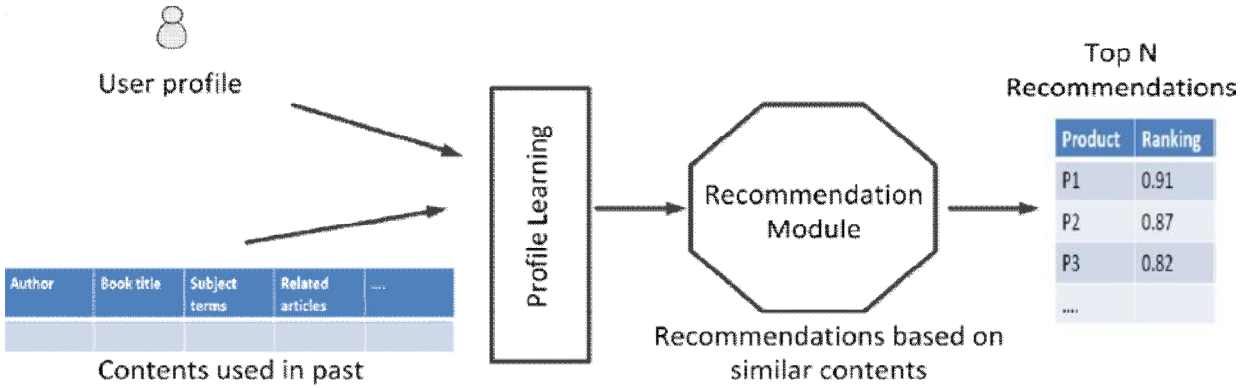
- **Term Frequency – Inverse Document Frequency(TF-IDF)**

- **Term Frequency**(문서에서 어떤 단어가 얼마나 자주 등장하는가)

- **Inverse Document Frequency**(그 단어가 다른 문서에서도 자주 등장하는가)

Inverse DF인 이유는 **다른 문서에서도 자주 등장하는 단어**라면 **중요도가 떨어지기 때문**이다.

영어에서 **The, That 같은 단어들은 어디에나 등장**하기 때문에 **Stop Word**로 따로 처리한다.



Content-based Filtering

- **TF-IDF 식**

$$W_{X,Y} = tf_{X,Y} * \log(\frac{N}{df_X})$$

문서 Y 에 있는 단어 X
tf = frequency of X in Y
df = number of documents containing X
N = total number of documents

- **Algorithm**

1) 먼저 알려진 **선호 Item 들의 집합에 대해 내용을 분석**
2) 사용자의 선호도를 나타내는 **User Profile 생성**
3) Item 과 User Profile사이의 **유사도를 계산**
4) User가 선호할 **다음 Item을 예측**(이때, 다른 Users의 데이터를 요구하지 않는다.)

Content-based Filtering

- **Example 1**

Tom plays soccer.(Doc 1)
Tom loves soccer and baseball. (Doc 2)
Baseball is his hobby and his job. (Doc 3)

<TF>

	Tom	Plays	Soccer	Loves	And	Baseball	Is	His	Hobby	Job
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

<IDF>

	Tom	Plays	Soccer	Loves	And	Baseball	Is	His	Hobby	Job
Doc 1	Log(3/2) = 0.18	Log(3/1) =0.48	Log(3/2) = 0.18							
Doc 2	0.18		0.18	0.48	0.18	0.18				
Doc 3					0.18	0.18	0.48	0.48	0.48	0.48

<TF-IDF>

	Tom	Plays	Soccer	Loves	And	Baseball	Is	His	Hobby	Job
Doc 1	0.18	0.48	0.18							
Doc 2	0.18		0.18	0.48	0.18	0.18				
Doc 3					0.18	0.18	0.48	0.96	0.48	0.48

Content-based Filtering

- **Example 2**

'IoT and analytics'를 검색해서 문서들의 총 Corpus는 10^6개라고 가정하면

Articles	Analytics	Data	Cloud	Smart	Insight
Article 1	21	24	0	2	2
Article 2	24	59	2	1	0
Article 3	40	115	8	10	19
Article 4	4	28	5	0	1
Article 5	8	48	4	3	4
Article 6	17	49	8	0	5
DF	5000	50000	10000	500000	7000

Content-based Filtering

TF값이 너무 커지는 걸 방지하기 위해 여기서는 $1 + \log(tf)$로 계산한다.

Articles	Analytics	Data	Cloud	Smart	Insight	Length of vector
Article 1	$TF = 1 + \log 21 = 2.322$	2.380	0	1.301	1.301	3.800
Article 2	2.380	2.770	1.301	1	0	4.004
Article 3	2.602	3.060	1.903	2	2.278	5.380
Article 4	1.602	2.447	1.698	0	1	3.527
Article 5	1.903	2.681	1.602	1.477	1.602	4.257
Article 6	2.230	2.690	1.903	0	1.698	4.326
IDF	2.301	1.301	2	0.301	2.154	

$\sqrt{2.322^2 + 2.380^2 + 0^2 + 1.301^2 + 1.301^2} = 3.800$

가장 자주 등장한 단어 'smart'의 IDF 값이 가장 낮은 걸 확인할 수 있다.

Content-based Filtering

Articles	Analytics	Data	Cloud	Smart	Insight	Sum of Normalized Lengths
Article 1	$\frac{2.322}{3.800} = 0.611$	0.626	0	0.342	0.342	1
Article 2	0.594	0.691	0.324	0.249	0	1
Article 3	0.483	0.568	0.353	0.371	0.423	1
Article 4	0.454	0.693	0.481	0	0.283	1
Article 5	0.447	0.629	0.376	0.346	0.376	1
Article 6	0.515	0.621	0.439	0	0.392	1

각 항목은 벡터로 저장되고 벡터 간의 유사성을 결정하기 위해 벡터 간의 각도를 계산한다.

Article Vector	Cosine Values
Cos(A1,A2)	0.880
Cos(A2,A3)	0.886
Cos(A1,A3)	0.922

0.611*0.594+0.626*0.691+0*0.324+0.342*0.249+0.342*0 = 0.880

IDF값은 중복되기 때문에 유사도를 측정할 때에는 배제한다.

Content-based Filtering

- ## 장점

  User-Item Interactions을 분석하는 것이 아니라 **컨텐츠 자체를 분석**하기 때문에 **프로세스가 덜 번거롭다.**

  Collaborative Filtering의 문제점인 **Cold Start가 없다.**

- ## 단점

  - 사용자 A가 B,C,D 카테고리에 관심이 있다고 판단되면 **그 외의 카테고리의 아이템은 추천하지 않는다.**
  - 새로운 사용자들은 기존의 정보가 없기 때문에 **Explicit 정보를 묻지 않는 한 추천이 힘들다.**

Hybrid Recommender Systems

- **Collaborative Filtering**
- **Content-Based Filtering**
- **Demographic Filtering**

사용자의 **인구통계학적 정보(Profile)를 기반으로 추천**을 제공한다.

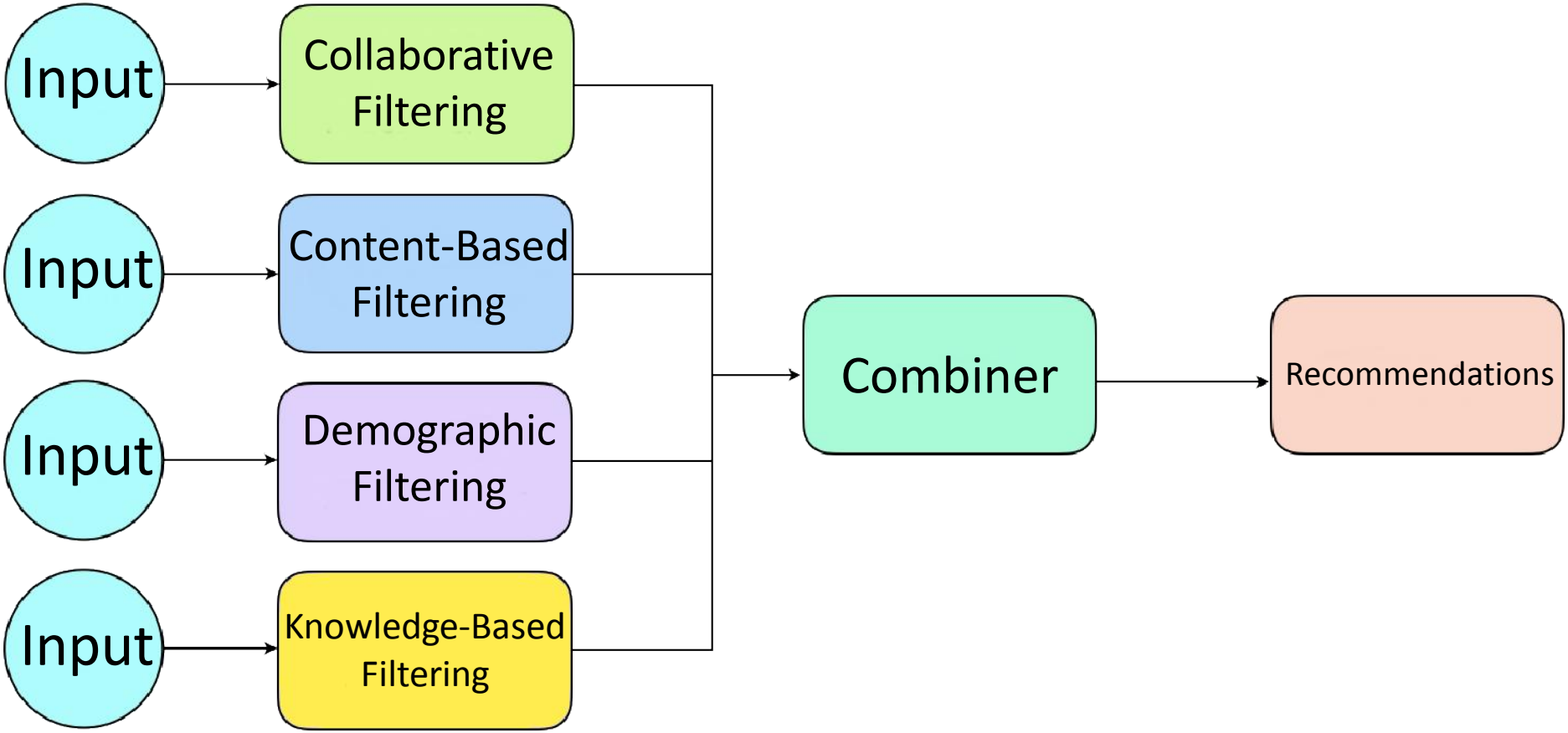
추천된 제품은 그 영역의 사용자들의 평가들을 합침으로써 **다른 인구통계학적 영역을 위해 만들어질 수 있다.**

- **Knowledge-based Filtering**

사용자의 선호와 요구(Needs)에 대한 추론을 기반으로 한 제품을 제안한다.

특정한 제품 특징이 사용자의 요구를 **얼마나 충족**시키는 지에 대한 뚜렷한 **기능적 지식을 포함**한다.

Hybrid Recommender Systems



Hybrid Recommender Systems

- **장점**
  - 신규 User에게는 CBF로 분석하여 추천하고, 충분한 데이터가 쌓인 후에는 CF로 추천의 정확성을 높아진다. ->**Cold Start Problem 해결**
  - **Sparsity Problem, Loss of Information, Knowledge Acquisition Bottleneck**과 같은 문제점을 극복할 수 있다.

- **단점**

복잡성이 증가하고 **구현 비용이 많이 든다.**