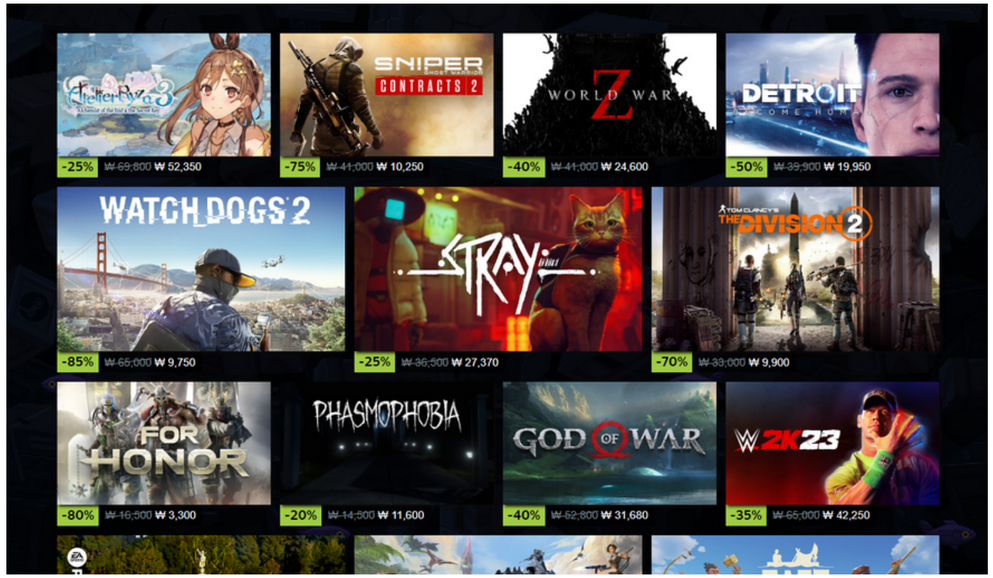
# 지식 그래프를 활용한 추천 서비스

방 학 1주 차

가천대학교 이다

# 추천 시스템이란?




↑ Steam Homepage

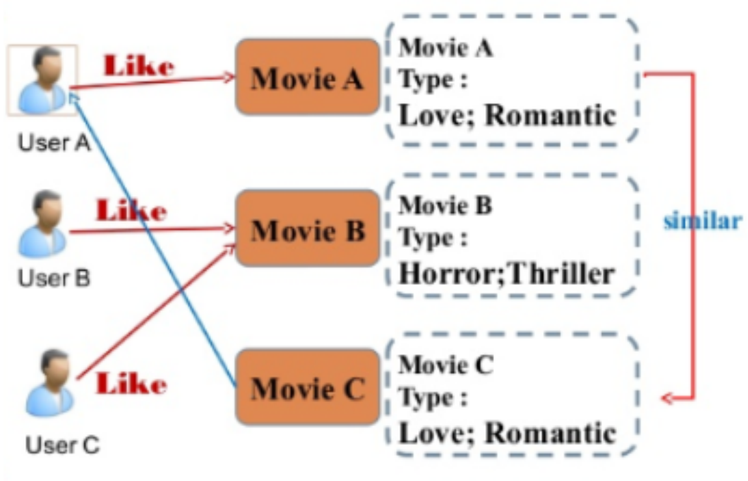
사용자의 구매, 플레이 데이터를 이용해서!

무엇을 기준으로 게임을 보여줄까?

# 추천 시스템의 종류 (Basic)

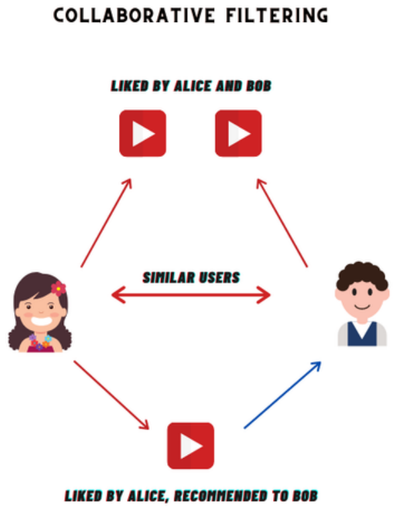
## Content-based Filtering

user와 item 중 item의 특성에 집중해서 추천함




## Collaborative Filtering

user와 item 중 user의 행동양식에 집중해서 추천함



# Content-based Filtering

## Content-based Filtering

Item Profile 만들기 → User Profile 만들기 → 유사도 비교 → 추천 진행

# Content-based Filtering

## Content-based Filtering
## Item Profile 만들기


### DOTA 2

- AOS 장르
- Valve 사가 유통함
- 12세 이용가


### Overwatch

- FPS
- Blizzard 사가 유통함
- 12세 이용가

# Content-based Filtering

## Content-based Filtering
## Item Profile 만들기


### DOTA 2

- AOS 장르
- Valve 사가 유통함
- 12세 이용가

⟶ 이걸 컴퓨터가 알아들을 수 있게 하려면?

**EMBEDDING!**

# Content-based Filtering

## Content-based Filtering
## Item Profile 만들기

## 임베딩을 하는 방식

(1) 글자 자체를 임베딩 하기

(2) 횟수를 기반으로 임베딩 하기

# Content-based Filtering

## Content-based Filtering
## Item Profile 만들기
## (1) 글자 자체를 임베딩 하기

### A. one-hot encoding

[판타지, 로맨스, 호러, 일상]

작품 A [1, 0, 0, 0]

작품 B [0, 1, 0, 0]

작품 C [0, 0, 1, 0]

작품 D [0, 0, 0, 1]

### B. Word2Vec

작품 E 설명 : Life of Students

작품 E : [0.245, 0.641, 0.364]

Life of Students

### C. Vocabulary

Vocab

Life … 1

of … 2

student … 3

작품 E : [1, 2, 3]

# Content-based Filtering

## Content-based Filtering
## Item Profile 만들기
## (2) 횟수를 기반으로 임베딩 하기

### A. CountVectorizer

문장 : This picture is awesome picture.

↓

BoW : [this : 0, picture : 1, is : 2, awesome : 3]

↓

CV : [1, 2, 1, 1]

### B. TF-IDF

여러 문장들

↓

TF-IDF를 이용한 가중치 계산

↓

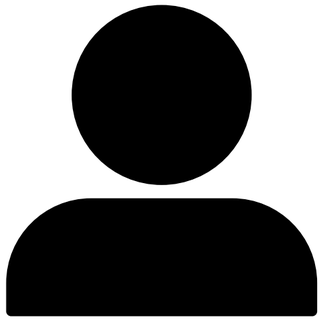
각 문장 별 Vector 생성!

# Content-based Filtering

## Content-based Filtering

## User Profile 만들기

### 홍길동 씨가 좋아하는 게임



→


**LOL**

- AOS 장르
- Riot 사가 유통함
- 12세 이용가

→

### Item Profile과 동일한 기준

↓

AOS FPS 스포츠 Bizzard 12세 이용가

**[1, 0, 0, 1, 0, 0, 1, 0]**

Riot Valve 청불

# Content-based Filtering

## Content-based Filtering

## 유사도 비교

item profile들과 user profile을  비교해서
가장 user랑 비슷한 item 찾기

# Content-based Filtering

## Content-based Filtering
## 유사도 비교

item profile들과 user profile을  비교해서
가장 user랑 비슷한 item 찾기

- Cosine Similarity
- Pearson Similarity
- Jaccard Similarity
- Manhattan Similiarity

# Content-based Filtering

## Content-based Filtering
## 유사도 비교
## Cosine Similarity

$$\cos(\theta) = \frac{a \cdot b}{|a||b|}$$

값의 범위 : -1 ~ 1 (0도면 1, 180도면 -1)

가장 기본적인 유사도 측정 방식

간단하게 사용하는 법 :

```
from sklearn.metrics.pairwise import cosine_similarity
```

# Content-based Filtering

## Content-based Filtering
## 유사도 비교
## Pearson Similarity

$$r = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum (x_i - \bar{x})^2 \sum (y_i - \bar{y})^2}}$$

값의 범위 : -1 ~ 1 (/ 형태면 1, \ 형태면 -1)

사용자가 극단적으로 점수를 주는 편일 때 사용

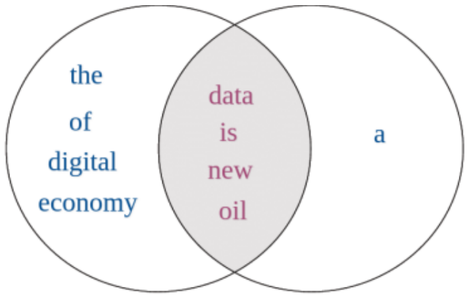
간단하게 사용하는 법 :

```
from scipy import stats
res = stats.pearsonr(x, y)
```

# Content-based Filtering

## Content-based Filtering
## 유사도 비교
## Jaccard Similarity

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




**값의 범위 : 0 ~ 1**

**sparse한 데이터에서도 사용하기 좋음**

**간단하게 사용하는 법 :**

```
a = set([1, 2, 3])
b = set([3, 4, 5])
gyo = a & b //교집합
hap = a | b //합집합
jaccard = len(gyo) / len(hap)
```

# Content-based Filtering

## Content-based Filtering

## 유사도 비교

## Manhattan Similarity

$$MaDistance = \sum_{i=1}^{n} |a_i - b_i|$$

값의 범위 : 0 ~

차원이 매우 높을 때 사용하기 좋음

간단하게 사용하는 법 :

```
from sklearn.metrics.pairwise import manhattan_distances
```

## Content-based Filtering

### 추천 진행

# 도출된 유사도 값들을 바탕으로
# 높은 유사도를 가진 것들을 추천함

## Content-based Filtering

## 장점

1. 다른 사용자의 데이터가 없어도 사용 가능함
2. 평점 등이 없어도 추천이 가능함
3. 추천을 하는 근거를 설명할 수 있음
4. 개인의 독특한 취향을 반영할 수 있음

## Content-based Filtering 단점

1. Feature을 뽑아내기 어려운 데이터들이 있음
2. Newbie 들에게는 추천을 하기가 어려움
3. 학습용 metadata를 필요로 함

# 과제

## Content-based Filtering

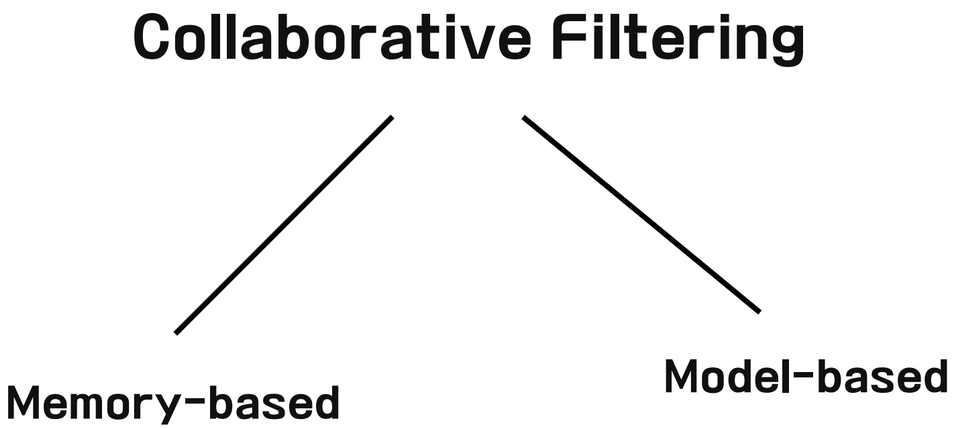
**과제 1**

지금까지 배운 내용들을 바탕으로
Content-based Filtering 코드를 작성해보기

**과제 2**

앞서 언급된 임베딩 / 유사도 기법들
써보고 비교해보기

# Collaborative Filtering



# Collaborative Filtering

## Collaborative Filtering

## (Memory Based)

사용자 간, 혹은 아이템 간의 관계를 통해 추천함.
(사용자가 아직 평가하지 않은 아이템을 예측하고자 함)

(1) 아이템 기반(item-based)

(2) 사용자 기반 (user-based)

# Collaborative Filtering

## Collaborative Filtering

## (Memory Based)

## User-based filtering

사용자들의 선호도를 기반으로 유사한 아이템을 추천

ex) 네이버 웹툰 아래에 뜨는
'OOO 독자님들이 좋아하는 웹툰'

# Collaborative Filtering

## Collaborative Filtering

## (Memory Based)

## item-based filtering

선호도가 많이 겹치는 사람이 선호하는 아이템을 추천

## Collaborative Filtering

## (Model Based)

사용자들과 아이템간의 숨어있는
특성값을 찾아내어 학습하는 방법

**=> 대표적인 모델 : Latent Factor Model**

# Collaborative Filtering

## Collaborative Filtering

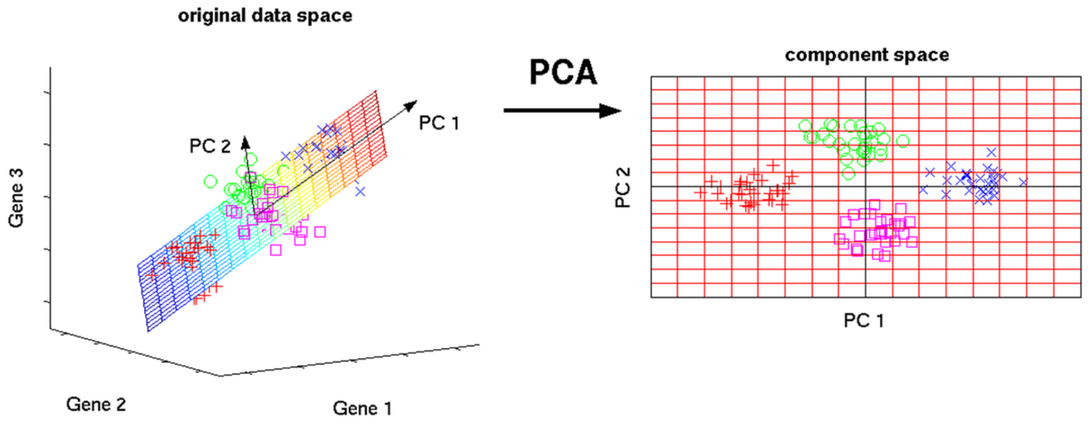
## (Latent Factor)

### SVD

$$A = U \cdot D \cdot V^T$$

- A : mxn rectangular matrix
- U : Left Singular Vectors (mxm orthogonal matrix)
- D : Singular Values (mxn diagonal matrix)
- V : Right Singular Vectors (nxn orthogonal matrix)
- T : Transpose matrix

### PCA



# Collaborative Filtering

## Collaborative Filtering

## (Latent Factor)

SVD, PCA 등을 통해
사용자 Latent와 유저 Latent로 분리

▼

두  행렬의 내적(행렬 곱)이
실제 평점과 유사해지도록 학습

## Collaborative Filtering

## 장점

1. 특별한 도메인 지식을 필요로 하지 않음

2. Metadata가 없어도 돌아감

# Collaborative Filtering

## Collaborative Filtering 단점

### Cold Start

새로운 아이템이
등장하면
추천이 곤란해짐

### Long Tail

추천 아이템이
관심을 많이 받는
소수의 아이템으로
쏠림

### No First-rater

아무도 평가하지
않은 아이템은
추천되지 않음

# 과제

## Collaborative Filtering

**과제 3**

지금까지 배운 내용들을 바탕으로
Collaborative Filtering 코드를 작성해보기

**과제 4**

https://tech.kakao.com/2021/10/18/collaborative-filtering/
'어떤 CF 모델을 선택할 것인가' 파트 읽고 정리해보기

# THANK YOU

CONTENT-BASED FILTERING & COLLABORATIVE FILTERING

가천대학교 이다