

Chapter 32. Recommendations



추천시스템?

추천시스템?

요즘은 추천시스템이 대부분 적용되어 있다

맞춤 동영상



[임진왜란1] 조선vs일본 전력 전격 분석, 조선이 초반에 털린 이유
황현필 한국사
조회수 6.1만회 • 1일 전



[스페셜] 대표 히트곡 부자 그룹! 자탄풍 히트곡 모음 슈가맨3(SUGARMAN3) 14회
JTBC Entertainment
조회수 10만회 • 1개월 전



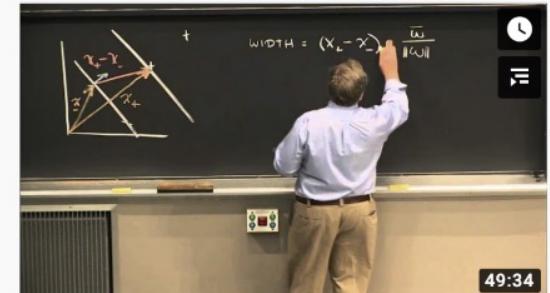
관련 재생목록 - 천상의 하모니♡ 정은지(Jung Eunji) '2020 너에게 난 나에게 난'♪ 슈가맨...
정은지, 편치(Punch), 천 등



[슈가송] 美친 고음의 끝판왕♪♪ 김상민 'You'♪ 투유 프로젝트 - 슈가맨2 2회
김상민, 자자(ZAZA), 더 크로스 등



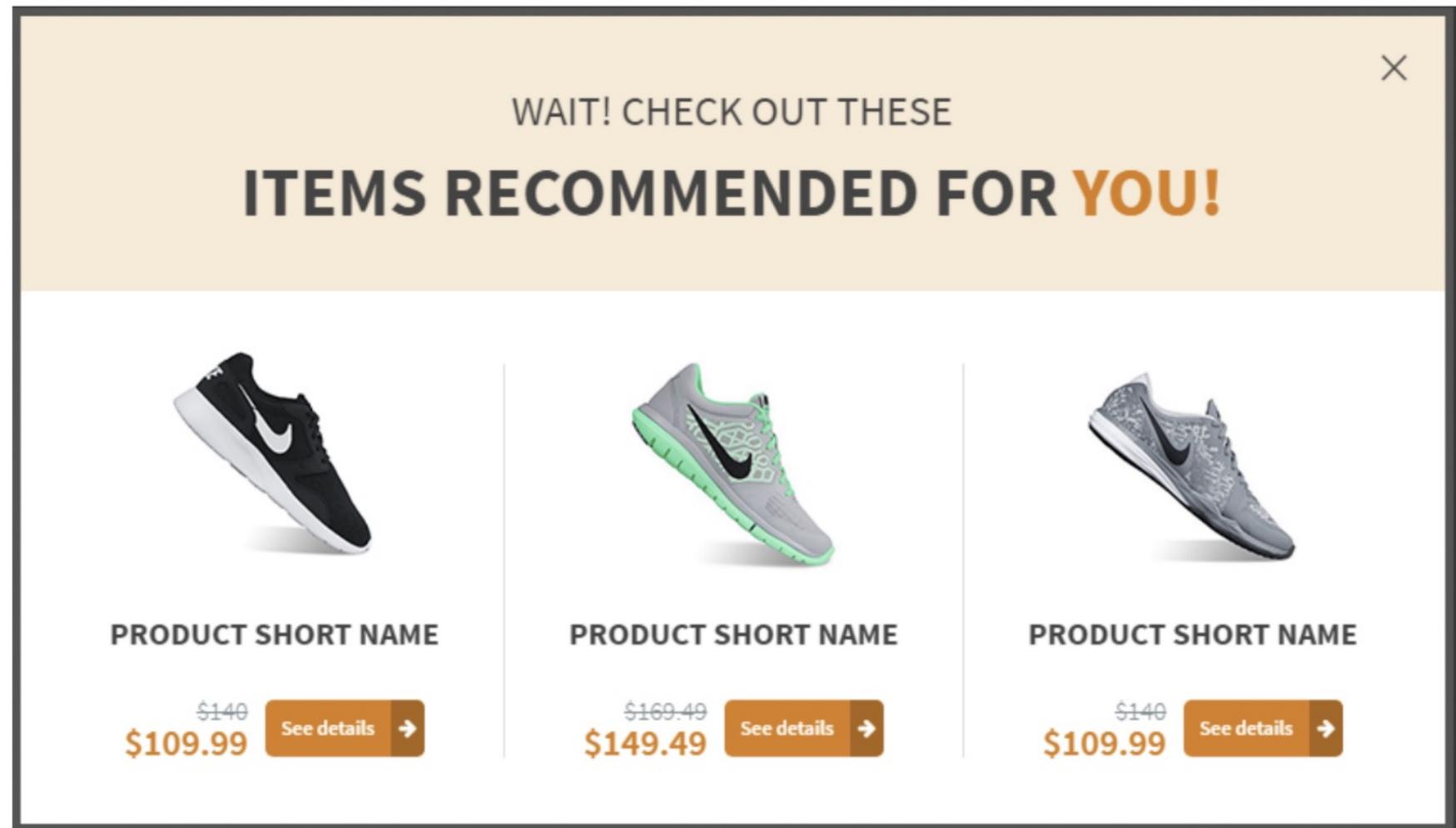
Autumn Gentle Pop Song
선선한 가을밤에 듣고 싶어지는 잔잔한 가을 팝송
Eric's Juke Box
조회수 109만회 • 6개월 전



16. Learning: Support Vector Machines
MIT OCW
MIT OpenCourseWare
조회수 102만회 • 6년 전

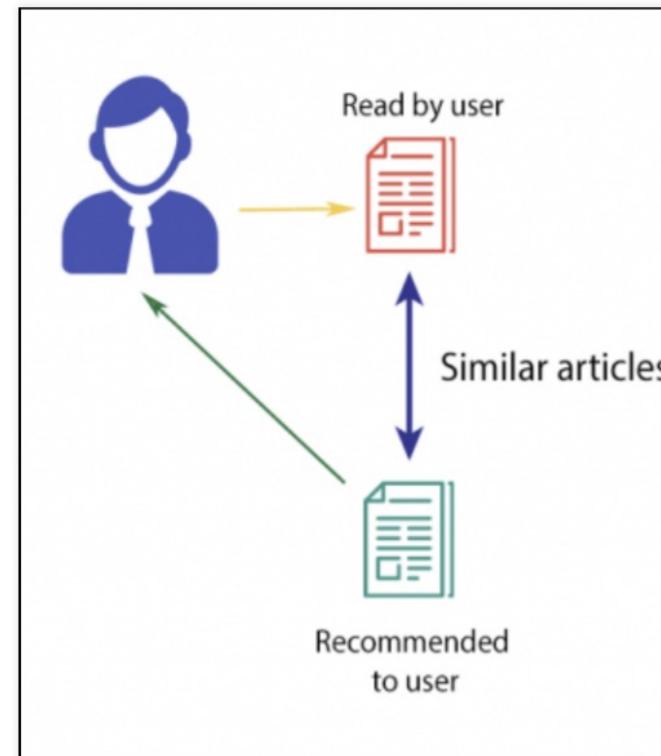
추천시스템?

특히 온라인 쇼핑몰 컨텐츠 등에서는 꽤 중요한 부분을 차지한다



추천시스템?

콘텐츠 기반 필터링 추천 시스템



- 사용자가 특정한 아이템을 선호하는 경우, 그 아이템과 비슷한 아이템을 추천하는 방식

추천시스템?

최근접 이웃 협업 필터링

- 축적된 사용자 행동 데이터를 기반으로 사용자가 아직 평가하지 않은 아이템을 예측 평가
- 사용자기반 : 당신과 비슷한 고객들이 다음 상품도 구매했음

	Item1	Item2	Item3	Item4	Item5
User A	5	4	4		
User B	5	3	4	5	3
User C	4	3	3	2	5

- 아이템기반 : 이 상품을 선택한 다른 고객들은 다음 상품도 구매했음

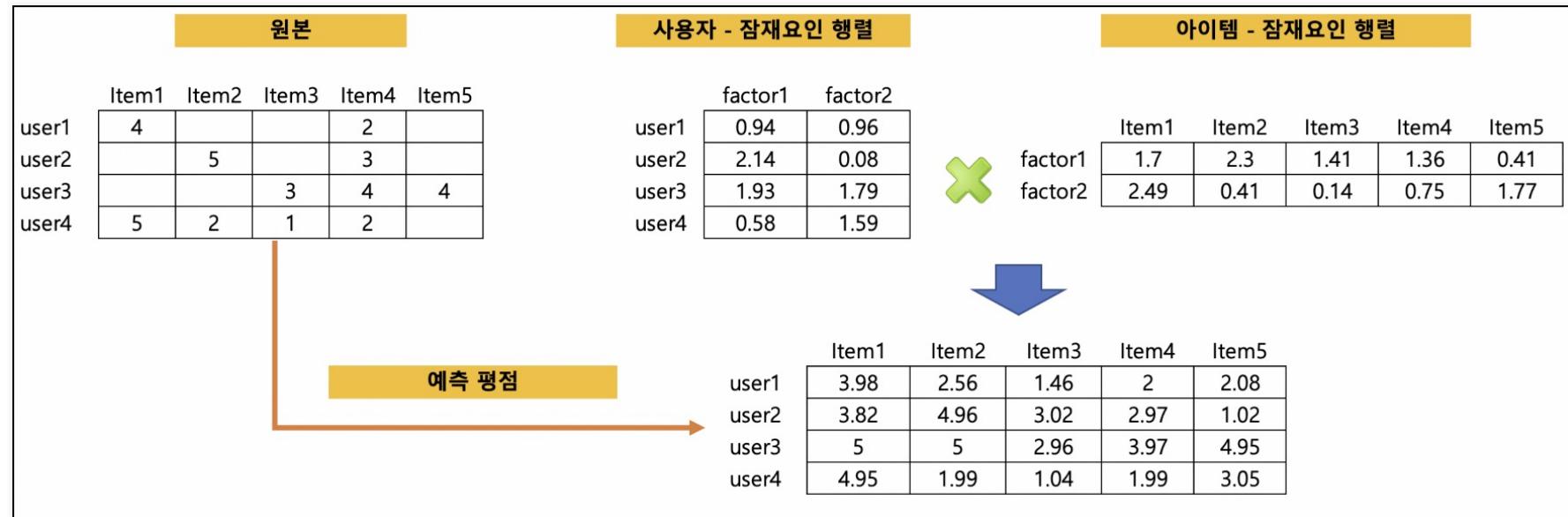
	User1	User2	User3	User4	User5
Item1	5	4	5	5	5
Item2	5	4	4		5
Item3	4	3	3		4

추천시스템?

- 일반적으로는 사용자 기반보다는 아이템 기반 협업 필터링이 정확도가 더 높음
 - 비슷한 영화를 좋아한다고 취향이 비슷하다고 판단하기 어렵거나
 - 매우 유명한 영화는 취향과 관계없이 관람하는 경우가 많고
 - 사용자들이 평점을 매기지 않는 경우가 많기 때문

추천시스템?

잠재 요인 협업 필터링



- 사용자-아이템 평점 행렬 데이터를 이용해서 “잠재요인”을 도출하는 것
- 주요인과 아이템에 대한 잠재요인에 대해 행렬 분해를 하고
다시 행렬곱을 통해 아직 평점을 부여하지 않은 아이템에 대한 예측 평점을 생성하는 것

실습

콘텐츠 기반 필터링 실습

- TMDB5000 영화 데이터 세트

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

데이터

The screenshot shows a Kaggle dataset page for the "TMDB 5000 Movie Dataset". The page features a collage of movie stills at the top. Below it, the title "TMDB 5000 Movie Dataset" and a subtitle "Metadata on ~5,000 movies from TMDb" are displayed. A "Dataset" icon is present. The TMDB logo and a note indicating the dataset was updated 3 years ago (Version 2) are also visible. The main navigation bar includes tabs for "Data" (which is selected), "Tasks (1)", "Kernels (1,513)", "Discussion (53)", "Activity", and "Metadata". To the right, there are buttons for "Download (44 MB)" and "New Notebook". A progress bar for a download is shown, with the number "2100" indicating the current count. Below the progress bar, there are sections for "Usability" (rating 8.2), "License" (Other, specified in description), and "Tags" (computing, arts and entertainment, film). A "Description" section is also present.

- <https://www.kaggle.com/tmdb/tmdb-movie-metadata>

데이터 읽기

```
import pandas as pd
import numpy as np
import warnings; warnings.filterwarnings('ignore')

movies =pd.read_csv('./tmdb-movie-metadata/tmdb_5000_movies.csv')
print(movies.shape)
movies.head()
```

(4803, 20)

	budget	genres	homepage	id	keywords	original_language	original_title
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "name": "..."}]	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id": ...}...]	en	Avatar
1	300000000	[{"id": 12, "name": "Adventure"}, {"id": 14, "name": "..."}]	http://disney.go.com/disneypictures/pirates/	285	[{"id": 270, "name": "ocean"}, {"id": 726, "name": "..."}...]	en	Pirates of the Caribbean: A World's End

데이터 선택

```
movies_df = movies[['id', 'title', 'genres', 'vote_average', 'vote_count',
                     'popularity', 'keywords', 'overview']]
movies_df.head()
```

		id	title	genres	vote_average	vote_count	popularity	keywords	overview
0	19995	Avatar		[{"id": 28, "name": "Action"}, {"id": 12, "nam...}	7.2	11800	150.437577	[{"id": 1463, "name": "culture clash"}, {"id": ...]	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End		[{"id": 12, "name": "Adventure"}, {"id": 14, "..."}	6.9	4500	139.082615	[{"id": 270, "name": "ocean"}, {"id": 726, "na...]	Captain Barbossa, long believed to be dead, ha...

- id, 영화제목 title, 장르 genres, 평균평점 vote_average, 투표수 vote_count, 인기 popularity, 키워드 keywords, 영화 개요 overview

데이터 주의 사항

```
| movies_df[['genres']][:1].values  
  
array([[[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"i  
d": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]]],  
      dtype=object)
```

- genres와 keywords는 컬럼안에 dict형으로 저장됨

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

문자열로 된 데이터를...

```
| from ast import literal_eval  
  
code = """(1, 2, {'foo': 'bar'})"""  
code
```

```
"(1, 2, {'foo': 'bar'})"
```

```
| type(code)
```

```
str
```

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

문자열로 된 데이터를 변환하자

```
| literal_eval(code)
```

```
(1, 2, {'foo': 'bar'})
```

```
| type(literal_eval(code))
```

```
tuple
```

genres와 keywords의 내용을 list와 dict으로 복구

```
from ast import literal_eval

movies_df[ 'genres' ] = movies_df[ 'genres' ].apply(literal_eval)
movies_df[ 'keywords' ] = movies_df[ 'keywords' ].apply(literal_eval)
movies_df.head()
```

			id	title	genres	vote_average	vote_count	popularity	keywords	overview
0	19995	Avatar			[{'id': 28, 'name': 'Action'}, {'id': 12, 'nam...}	7.2	11800	150.437577	[{'id': 1463, 'name': 'culture clash'}, {'id': ...]	In the 22nd century, a paraplegic Marine is di...
1	285	Pirates of the Caribbean: At World's End			[{'id': 12, 'name': 'Adventure'}, {'id': 14, '...}	6.9	4500	139.082615	[{'id': 270, 'name': 'ocean'}, {'id': 726, 'na...]	Captain Barbosa, long believed to be dead, ha...

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

dict의 value 값을 특성으로 사용하도록 변경

```
movies_df['genres'] = movies_df['genres'].apply(lambda x : [y['name'] for y in x])
movies_df['keywords'] = movies_df['keywords'].apply(lambda x : [y['name'] for y in x])
movies_df[['genres', 'keywords']][:2]
```

	genres	keywords
0	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...]
1	[Adventure, Fantasy, Action]	[ocean, drug abuse, exotic island, east india ...]

genres의 각 단어들을 하나의 문장(띄어쓰기로 구분된)으로 변환

```
movies_df['genres_literal'] = movies_df['genres'].apply(lambda x : (' ').join(x))  
movies_df.head()
```

		id	title	genres	vote_average	vote_count	popularity	keywords	overview	genres_literal
0	19995	Avatar	Avatar	[Action, Adventure, Fantasy, Science Fiction]	7.2	11800	150.437577	[culture clash, future, space war, space colon...]	In the 22nd century, a paraplegic Marine is di...	Action Adventure Fantasy Science Fiction
1	285	Pirates of the Caribbean: At World's End	Pirates of the Caribbean: At World's End	[Adventure, Fantasy, Action]	6.9	4500	139.082615	[ocean, drug abuse, exotic island, east india ...]	Captain Barbossa, long believed to be dead, ha...	Adventure Fantasy Action

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

문자열로 변환된 genres를 CountVectorizer 수행

```
| from sklearn.feature_extraction.text import CountVectorizer  
  
count_vect = CountVectorizer(min_df=0, ngram_range=(1,2))  
genre_mat = count_vect.fit_transform(movies_df['genres_literal'])  
print(genre_mat.shape)
```

(4803, 276)

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

코사인 유사도



콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

코사인 유사도 계산식

$$\text{similarity} = \cos(\Theta) = \frac{A \cdot B}{||A|| ||B||} = \frac{\sum_{i=1}^n A_i \times B_i}{\sqrt{\sum_{i=1}^n (A_i)^2} \times \sqrt{\sum_{i=1}^n (B_i)^2}}$$

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

문장의 유사도 측정을 하는 방법 중 하나인 코사인 유사도 측정을 수행

```
| from sklearn.metrics.pairwise import cosine_similarity  
  
genre_sim = cosine_similarity(genre_mat, genre_mat)  
print(genre_sim.shape)  
print(genre_sim[:2])  
  
(4803, 4803)  
[[1.          0.59628479 0.4472136   ... 0.          0.          0.          ]  
 [0.59628479 1.          0.4          ... 0.          0.          0.          ]]
```

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

confusion_matrix와 비슷하게 해석하면 된다

	item0	item1	item2	item3
item0	1	7	2	
item1	1	2	4	
item2	0	8	3	
item3	2	0	3	

cosine 유사도 →

	item0	item1	item2	item3
item0	1	0.68	0.99	0.3
item1	0.68	1	0.72	0.85
item2	0.99	0.72	1	0.29
item3	0.3	0.85	0.29	1

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

genre_sim 객체에서 높은 값 순으로 정렬할 수 있다

```
| genre_sim_sorted_ind = genre_sim.argsort()[:, ::-1]
| print(genre_sim_sorted_ind[:1])
```

```
[[    0 3494  813 ... 3038 3037 2401]]
```

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

추천 영화를 DataFrame으로 반환하는 함수

```
| def find_sim_movie(df, sorted_ind, title_name, top_n=10):
|     title_movie = df[df['title'] == title_name]
|
|     title_index = title_movie.index.values
|     similar_indexes = sorted_ind[title_index, :(top_n)]
|
|     print(similar_indexes)
|     similar_indexes = similar_indexes.reshape(-1)
|
|     return df.iloc[similar_indexes]
```

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

영화 대부와 유사한 영화는?



뭘까?

```
similar_movies = find_sim_movie(movies_df, genre_sim_sorted_ind, 'The Godfather', 10)
similar_movies[['title', 'vote_average']]
```

```
[[2731 1243 3636 1946 2640 4065 1847 4217 883 3866]]
```

		title	vote_average
2731	The Godfather: Part II		8.3
1243	Mean Streets		7.2
3636	Light Sleeper		5.7
1946	The Bad Lieutenant: Port of Call - New Orleans	6.0	
2640	Things to Do in Denver When You're Dead	6.7	
4065	Mi America	0.0	
1847	GoodFellas	8.2	
4217	Kids	6.8	
883	Catch Me If You Can	7.7	
3866	City of God	8.1	

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

뭐 그런 것도 같고



- 다시 전 화면으로 가보자.. 뭔가 문제들이 있다.

다시 데이터로 돌아가보자

```
movies_df[['title', 'vote_average', 'vote_count']].sort_values('vote_average',  
ascending=False)[ :10]
```

		title	vote_average	vote_count
3519	Stiff Upper Lips		10.0	1
4247	Me You and Five Bucks		10.0	2
4045	Dancer, Texas Pop. 81		10.0	1
4662	Little Big Top		10.0	1
3992	Sardaarji		9.5	2
2386	One Man's Hero		9.3	2
2970	There Goes My Baby		8.5	2
1881	The Shawshank Redemption		8.5	8205
2796	The Prisoner of Zenda		8.4	11
3337	The Godfather		8.4	5893

- 평점과 평점을 매긴 횟수를 보면 문제 데이터가 보인다.

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

영화 선정을 위한 가중치를 선정해보자

$$\left(\frac{v}{v+m}\right)R + \left(\frac{m}{v+m}\right)C$$

- v : 개별 영화에 평점을 투표한 횟수
- m : 평점을 부여하기 위한 최소 투표 횟수
- R : 개별 영화에 대한 평균 평점
- C : 전체 영화에 대한 평균 평점

콘텐츠 기반 필터링
실습 - TMDB5000
영화 데이터 세트

영화 전체 평균평점과 최소 투표 횟수를 60%지점으로 지정

```
C = movies_df['vote_average'].mean()
m = movies_df['vote_count'].quantile(0.6)
print('C:', round(C, 3), 'm:', round(m, 3))
```

C: 6.092 m: 370.2

가중치가 부여된 평점을 계산하기 위한 함수

```
| def weighted_vote_average(record):  
|     v = record[ 'vote_count' ]  
|     R = record[ 'vote_average' ]  
  
|     return ( (v/(v+m)) * R ) + ( (m/(m+v)) * C )
```

다시 계산

```
movies_df['weighted_vote'] = movies_df.apply(weighted_vote_average, axis=1)
movies_df.head()
```

		id	title	genres	vote_average	vote_count	popularity	keywords	overview	genres_literal	weighted_vote
		19995	Avatar	[Action, Adventure, Fantasy, Science Fiction]	7.2	11800	150.437577	[culture clash, future, space war, space colon...]	In the 22nd century, a paraplegic Marine is di...	Action Adventure Fantasy Science Fiction	7.166301
		285	Pirates of the Caribbean: At World's End	[Adventure, Fantasy, Action]	6.9	4500	139.082615	[ocean, drug abuse, exotic island, east india ...]	Captain Barbossa, long believed to be dead, ha...	Adventure Fantasy Action	6.838594

전체 데이터에서 가중치가 부여된 평점 순으로 정렬한 결과

```
movies_df[['title','vote_average',
           'weighted_vote','vote_count']].sort_values('weighted_vote',
                                                       ascending=False)[:10]
```

		title	vote_average	weighted_vote	vote_count
1881	The Shawshank Redemption	8.5	8.396052	8205	
3337	The Godfather	8.4	8.263591	5893	
662	Fight Club	8.3	8.216455	9413	
3232	Pulp Fiction	8.3	8.207102	8428	
65	The Dark Knight	8.2	8.136930	12002	
1818	Schindler's List	8.3	8.126069	4329	
3865	Whiplash	8.3	8.123248	4254	
809	Forrest Gump	8.2	8.105954	7927	
2294	Spirited Away	8.3	8.105867	3840	
2731	The Godfather: Part II	8.3	8.079586	3338	

유사 영화를 찾는 함수를 변경

```
def find_sim_movie(df, sorted_ind, title_name, top_n=10):
    title_movie = df[df['title'] == title_name]
    title_index = title_movie.index.values

    similar_indexes = sorted_ind[title_index, :(top_n*2)]
    similar_indexes = similar_indexes.reshape(-1)

    similar_indexes = similar_indexes[similar_indexes != title_index]

    return df.iloc[similar_indexes].sort_values('weighted_vote',
                                                ascending=False)[:top_n]
```

다시 대부와 유사한 영화 찾기

```
similar_movies = find_sim_movie(movies_df, genre_sim_sorted_ind, 'The Godfather', 10)
similar_movies[['title', 'vote_average', 'weighted_vote']]
```

		title	vote_average	weighted_vote
2731	The Godfather: Part II	8.3	8.079586	
1847	GoodFellas	8.2	7.976937	
3866	City of God	8.1	7.759693	
1663	Once Upon a Time in America	8.2	7.657811	
883	Catch Me If You Can	7.7	7.557097	
281	American Gangster	7.4	7.141396	
4041	This Is England	7.4	6.739664	
1149	American Hustle	6.8	6.717525	
1243	Mean Streets	7.2	6.626569	
2839	Rounders	6.9	6.530427	

아이템 기반 최근접 이웃 협업 필터링

데이터

The screenshot shows the GroupLens website with a blue header bar containing the logo and navigation links for 'about', 'datasets', 'publications', and 'blog'. The main content area has a light gray background. On the left, a large heading 'MovieLens Latest Datasets' is displayed in a teal font. Below it, a paragraph explains that the datasets will change over time and are not appropriate for reporting research results. It mentions two datasets: 'Small' (100,000 ratings and 3,600 tag applications applied to 9,000 movies by 600 users, last updated 9/2018) and 'Full' (27,000,000 ratings and 1,100,000 tag applications applied to 58,000 movies by 280,000 users, includes tag genome data with 14 million relevance scores across 1,100 tags, last updated 9/2018). Each dataset entry includes a 'README.html' link and a download link for a ZIP file. A 'Permalink' is provided at the bottom. On the right side, there is a sidebar titled 'Datasets' with a list of other datasets: MovieLens (selected), WikiLens, Book-Crossing, Jester, EachMovie, HetRec 2011, and Serendipity 2018.

- <https://grouplens.org/datasets/movielens/latest/>
- 영화의 평점을 매긴 사용자와 영화 평점 행렬 등의 데이터
- 이중에서 1MB짜리 small 데이터를 받자 - 정신건강을 위해...

아이템 기반 최근접
이웃 협업 필터링

데이터 읽기

```
| import pandas as pd
| import numpy as np
|
| movies = pd.read_csv('./ml-latest-small/movies.csv')
| ratings = pd.read_csv('./ml-latest-small/ratings.csv')
| print(movies.shape)
| print(ratings.shape)
```

```
(9742, 3)
(100836, 4)
```

아이템 기반 최근접
이웃 협업 필터링

movie에는 영화 제목과 장르가

```
movies.head()
```

	movied	title	genres
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	2	Jumanji (1995)	Adventure Children Fantasy
2	3	Grumpier Old Men (1995)	Comedy Romance
3	4	Waiting to Exhale (1995)	Comedy Drama Romance
4	5	Father of the Bride Part II (1995)	Comedy

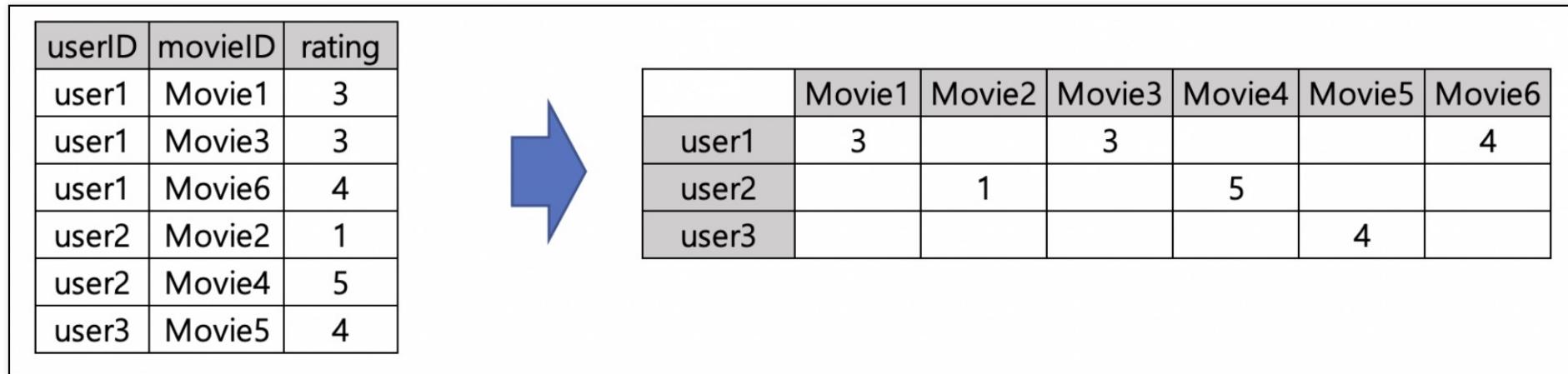
ratings에는 영화 평점이 사용자별로 영화별로 존재한다

```
ratings.head()
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

아이템 기반 최근접
이웃 협업 필터링

raw 데이터를 정리해야 한다



The diagram illustrates the transformation of raw data from a user-item rating matrix to a user-item rating matrix where items are columns.

Raw Data (Left):

userID	movieID	rating
user1	Movie1	3
user1	Movie3	3
user1	Movie6	4
user2	Movie2	1
user2	Movie4	5
user3	Movie5	4

Transformed Data (Right):

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6
user1	3		3			4
user2		1		5		
user3					4	

- 이때 사용하는 magical command는??

아이템 기반 최근접
이웃 협업 필터링

pivot_table

```
ratings = ratings[['userId', 'movieId', 'rating']]  
ratings_matrix = ratings.pivot_table('rating', index='userId', columns='movieId')  
ratings_matrix.head()
```

movieId	1	2	3	4	5	6	7	8	9	10	...	193565	193567	193571	193573	193579	193581
userId																	
1	4.0	NaN	4.0	NaN	NaN	4.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN									
3	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN									
4	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN									
5	4.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN								

5 rows x 9724 columns

ratings와 movie를 movieID 기준으로 결합

```
rating_movies = pd.merge(ratings, movies, on='movieId')
rating_movies.head()
```

	userId	movieId	rating	title	genres
0	1	1	4.0	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
1	5	1	4.0	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
2	7	1	4.5	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
3	15	1	2.5	Toy Story (1995)	Adventure Animation Children Comedy Fantasy
4	17	1	4.5	Toy Story (1995)	Adventure Animation Children Comedy Fantasy

아이템 기반 최근접
이웃 협업 필터링

그럼 다시 pivot_table을 이용해서 정리

```
ratings_matrix = rating_movies.pivot_table('rating', index='userId', columns='title')  
ratings_matrix.head()
```

userId	title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[R] (2014)
1		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
2		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
3		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
4		NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN

아이템 기반 최근접
이웃 협업 필터링

nan은 0으로 변환

```
ratings_matrix = ratings_matrix.fillna(0)
ratings_matrix.head()
```

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)	...	Zulu (2013)	[R (2013)]
userId													
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0

유사도 측정을 위해 행렬의 transpose

```
ratings_matrix_T = ratings_matrix.transpose()  
ratings_matrix_T.head()
```

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
	title																				
'71 (2014)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	4.0
'Hellboy': The Seeds of Creation (2004)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
'Round Midnight (1986)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
'Salem's Lot (2004)	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

유사도 측정 결과

```
from sklearn.metrics.pairwise import cosine_similarity

item_sim = cosine_similarity(ratings_matrix_T, ratings_matrix_T)
item_sim_df = pd.DataFrame(data=item_sim, index=ratings_matrix.columns,
                           columns=ratings_matrix.columns)
print(item_sim_df.shape)
item_sim_df.head()
```

(9719, 9719)

title	'71 (2014)	'Hellboy': The Seeds of Creation (2004)	'Round Midnight (1986)	'Salem's Lot (2004)	'Til There Was You (1997)	'Tis the Season for Love (2015)	'burbs, The (1989)	'night Mother (1986)	(500) Days of Summer (2009)	*batteries not included (1987)
title												
'71 (2014)	1.0	0.000000	0.000000	0.000000	0.000000	0.0	0.000000	0.0	0.141653	0.0	...	0.0
'Hellboy': The Seeds of Creation (2004)	0.0	1.000000	0.707107	0.000000	0.000000	0.0	0.000000	0.0	0.000000	0.0	...	0.0

대부와 유사한 영화는?

```
item_sim_df[ "Godfather, The (1972)" ].sort_values(ascending=False)[ :6 ]
```

title	
Godfather, The (1972)	1.000000
Godfather: Part II, The (1974)	0.821773
Goodfellas (1990)	0.664841
One Flew Over the Cuckoo's Nest (1975)	0.620536
Star Wars: Episode IV - A New Hope (1977)	0.595317
Fargo (1996)	0.588614

Name: Godfather, The (1972), dtype: float64

아이템 기반 최근접
이웃 협업 필터링

영화 인셉션



인셉션과 유사한 영화

```
| item_sim_df["Inception (2010)"].sort_values(ascending=False)[1:6]
```

```
title
Dark Knight, The (2008)      0.727263
Inglourious Basterds (2009)  0.646103
Shutter Island (2010)        0.617736
Dark Knight Rises, The (2012) 0.617504
Fight Club (1999)            0.615417
Name: Inception (2010), dtype: float64
```