MovieLens data – Item based CF with dot production

5조 - 인생이 핀다

빅데이터처리및응용

신우현, 윤서환, 윤성호

20192040, 2019204045, 2019204023

광운대학교
KwangWoon University

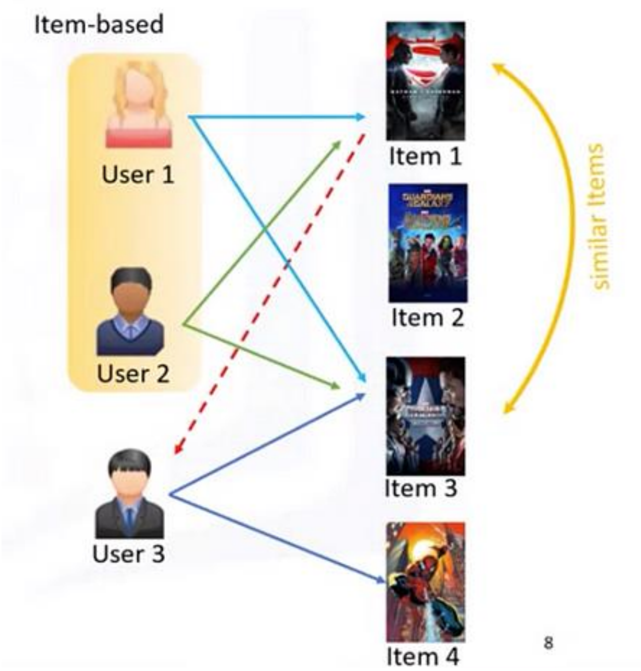
Outline

1. Introduction
   - 1.1 Project goal
   - 1.2 Pipeline
2. Data preprocessing
   - 2.1 Genre missing value
   - 2.2 Title duplicated
   - 2.3 Dataset
3. Item – genre
   - 3.1 Baseline
   - 3.2 TF-IDF
4. Item - user
   - 4.1 Baseline
   - 4.2 KNN
   - 4.3 SVD
5. Result
   - 5.1 How to use
   - 5.2 Limitation

KWANGWOON UNIVERSITY

Introduction

- **Project goal**
  - MovieLens dataset
  - Item based CF with dot production

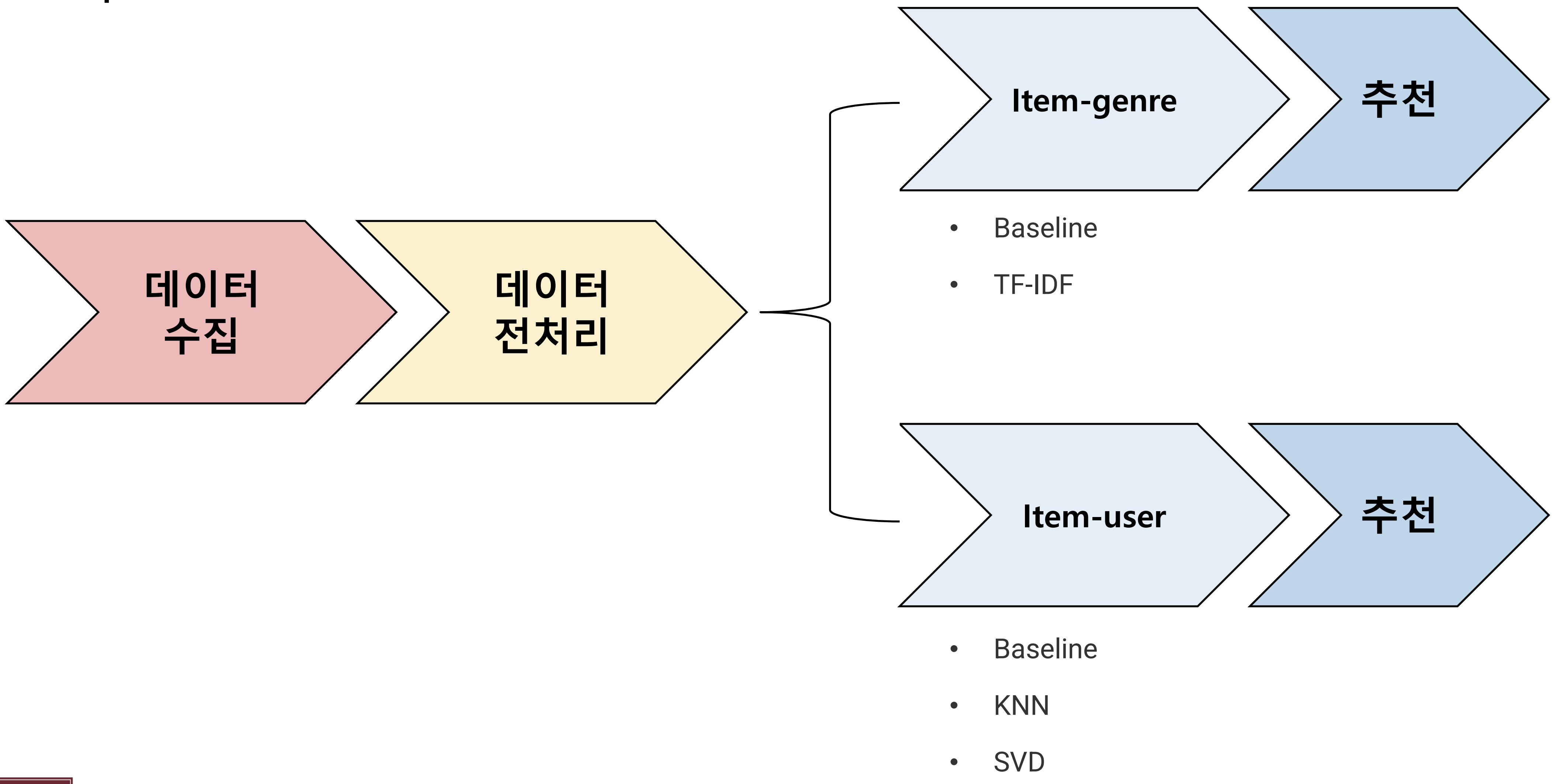


다양한 접근 ➡ 최적의 추천

https://medium.com/@toprak.mhmt/collaborative-filtering-3ceb89080ade

Introduction

- **Pipeline**



Outline

KWANGWOON UNIVERSITY

Data preprocessing

- **Genre missing value**
  - IMDB에서 Crawling을 통한 결측 대체
  - KNN 시도 → domain 지식 이용

IMDb

Crawling

	title	genres	imdbId	tmdbId
movieId				
114335	La cravate (1957)	Short|Comedy	0121731	32891.0
122888	Ben-hur (2016)	Action|Drama|Romance	2638144	271969.0
122896	Pirates of the Caribbean: Dead Men Tell No Tal...	Action|Adventure|Fantasy	1790809	166426.0
129250	Superfast! (2015)	Comedy	2933474	325358.0
132084	Let It Be Me (1995)	Drama|Romance	0113638	335145.0
134861	Trevor Noah: African American (2013)	Comedy	3043546	250556.0
141131	Guardians (2016)	Action|Adventure|Comedy	4600952	354556.0
141866	Green Room (2015)	Crime|Drama|Horror	4062536	313922.0
142456	The Brand New Testament (2015)	Comedy|Fantasy	3792960	330764.0
143410	Hyena Road	Action|Drama|War	4034452	316042.0
147250	The Adventures of Sherlock Holmes and Doctor W...	Crime|Drama|Mystery	0229922	127605.0
149330	A Cosmic Christmas (1977)	Animation|Family|Sci-Fi	0182015	125464.0
152037	Grease Live (2016)	Comedy|Musical|Romance	4366830	348089.0
155589	Noin 7 veljestä (1968)	Adventure|Comedy	0134854	55495.0
156605	Paterson	Comedy|Drama|Romance	5247022	370755.0
159161	Ali Wong: Baby Cobra (2016)	Comedy	5066574	396292.0
159779	A Midsummer Night's Dream (2016)	Comedy|Fantasy|Romance	5051278	398854.0
161008	The Forbidden Dance (1990)	Drama|Music|Romance	0099595	118430.0
165489	Ethel & Ernest (2016)	Animation|Drama|History	1725969	413770.0
166024	Whiplash (2013)	Short|Drama|Music	2654430	367412.0
167570	The OA	Drama|Fantasy|Mystery	4635282	432192.0
169034	Lemonade (2016)	Music|Musical	5662106	394269.0
171495	Cosmos	Documentary	0081846	409926.0
171631	Maria Bamford: Old Baby	Documentary|Comedy	6264596	455601.0
171749	Death Note: Desu nôto (2006–2007)	Animation|Crime|Drama	0877057	419787.0
171891	Generation Iron 2	Documentary	6263642	447818.0
172497	T2 3-D: Battle Across Time (1996)	Short|Action|Sci-Fi	0117880	65595.0
172591	The Godfather Trilogy: 1972-1990 (1992)	Crime|Drama|Thriller	0150742	364150.0
173535	The Adventures of Sherlock Holmes and Doctor W...	Crime|Drama|Mystery	0459945	406403.0
174403	The Putin Interviews (2017)	Documentary|Biography	6840134	461805.0
176601	Black Mirror	Short	2492564	452830.0
181413	Too Funny to Fail: The Life and Death of The D...	Documentary	7544820	482004.0
181719	Serving in Silence: The Margarethe Cammermeyer...	Biography|Drama	0114395	49809.0
182727	A Christmas Story Live! (2017)	Musical	6881890	485517.0

https://data-science-hi.tistory.com/82

Data preprocessing

- **Title duplicated**
  - Baseline code에서 title 중복 발생
  - 연도 제거에 따른 중복 title 발생

```
movieId    9703
title      9428
year        106
genres      950
dtype: int64
```

```
Hamlet                                                          5
Jane Eyre                                                       4
Misérables, Les                                                 4
Christmas Carol, A                                              4
Three Musketeers, The                                           4
                                                               ..
All That Heaven Allows                                          1
Barefoot Contessa, The                                          1
Cries and Whispers (Viskningar och rop)                         1
Garden of the Finzi-Continis, The (Giardino dei Finzi-Contini, Il)    1
Andrew Dice Clay: Dice Rules                                    1
Name: title, Length: 9428, dtype: int64
```

Data preprocessing

- **Title duplicated**

	movieId	title	genres	year
6	7	Sabrina1	Comedy Romance	1995
27	28	Persuasion1	Drama Romance	1995
65	73	Misérables, Les1	Drama War	1995
79	88	Black Sheep1	Comedy	1996
84	95	Broken Arrow1	Action Adventure Thriller	1996
...	...	...	...	...
9601	176413	Bliss2	Drama	2012
9615	177763	Murder on the Orient Express2	Crime Drama Mystery	2017
9686	184349	Elsa & Fred2	Comedy Drama Romance	2005
9691	184931	Death Wish2	Action Crime Drama Thriller	2018
9714	189111	Spiral2	Documentary	2018

527 rows × 4 columns

Title + 숫자

↓

중복 데이터 전처리

Data preprocessing

- **Dataset**

	userId	movieId	rating	title	genres	year
0	1	1	4.0	Toy Story	Adventure Animation Children Comedy Fantasy	1995
1	1	3	4.0	Grumpier Old Men	Comedy Romance	1995
2	1	6	4.0	Heat	Action Crime Thriller	1995
3	1	47	5.0	Seven (a.k.a. Se7en)	Mystery Thriller	1995
4	1	50	5.0	Usual Suspects, The	Crime Mystery Thriller	1995
...	...	...	...	...	...	...
100831	610	166534	4.0	Split	Drama Horror Thriller	2017
100832	610	168248	5.0	John Wick: Chapter Two	Action Crime Thriller	2017
100833	610	168250	5.0	Get Out	Horror	2017
100834	610	168252	5.0	Logan	Action Sci-Fi	2017
100835	610	170875	3.0	The Fate of the Furious	Action Crime Drama Thriller	2017

100836 rows × 6 columns

(100836 x 6)

Dataset 구축

Outline

KWANGWOON UNIVERSITY

Item - genre

- **Baseline**
  - 장르 존재 유무 → one hot encoding
  - Pearson, Cosine 유사도 측정

title	Documentary	Adventure	IMAX	History	Action	Family	Drama	Musical	Fantasy	Music	...	Sci-Fi	Biography
Sabrina1	0	0	0	0	0	0	0	0	0	0	...	0	0
Persuasion1	0	0	0	0	0	0	1	0	0	0	...	0	0
Misérables, Les1	0	0	0	0	0	0	1	0	0	0	...	0	0
Black Sheep1	0	0	0	0	0	0	0	0	0	0	...	0	0
Broken Arrow1	0	1	0	0	1	0	0	0	0	0	...	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
Black Butler: Book of the Atlantic	0	0	0	0	1	0	0	0	1	0	...	0	0
No Game No Life: Zero	0	0	0	0	0	0	0	0	1	0	...	0	0
Flint	0	0	0	0	0	0	1	0	0	0	...	0	0
Bungo Stray Dogs: Dead Apple	0	0	0	0	1	0	0	0	0	0	...	0	0
Andrew Dice Clay: Dice Rules	0	0	0	0	0	0	0	0	0	0	...	0	0

9737 rows × 24 columns

Item - genre

- **Baseline – Pearson**
  - Pearson 유사도 계산

```
# 피어슨 상관계수를 계산하여 유사도 행렬 생성
pearson_similarity = movie_genre.T.corr(method='pearson')
```

	userId	title	pred_rating
0	1	Sabrina1	4.080404
1	1	Persuasion1	4.421699
2	1	Misérables, Les1	4.549232
3	1	Black Sheep1	4.149004
4	1	Broken Arrow1	4.222394
...	...	...	...
9732	610	Black Butler: Book of the Atlantic	3.608077
9733	610	No Game No Life: Zero	3.703827
9734	610	Flint	3.941182
9735	610	Bungo Stray Dogs: Dead Apple	3.531496
9736	610	Andrew Dice Clay: Dice Rules	3.714786

5939570 rows × 3 columns

```
MAE  : 0.7458
MSE  : 0.94524
RMSE : 0.97224
```

Item - genre

- **Baseline – Cosine**
  - Cosine 유사도 계산

```
from sklearn.metrics.pairwise import cosine_similarity

# 코사인 유사도 행렬
movie_similarity_matrix = cosine_similarity(movie_genre)
```

	userId	title	pred_rating
0	1	Sabrina1	4.206213
1	1	Persuasion1	4.374184
2	1	Misérables, Les1	4.423022
3	1	Black Sheep1	4.209127
4	1	Broken Arrow1	4.270043
...	...	...	...
9732	610	Black Butler: Book of the Atlantic	3.649271
9733	610	No Game No Life: Zero	3.689688
9734	610	Flint	3.863857
9735	610	Bungo Stray Dogs: Dead Apple	3.615727
9736	610	Andrew Dice Clay: Dice Rules	3.703244

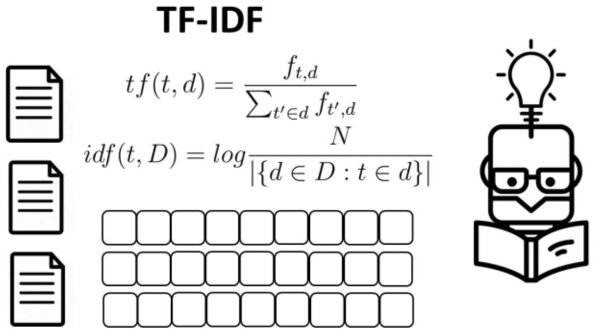
5939570 rows × 3 columns

Pearson 대비 성능 향상

```
MAE  : 0.70819
MSE  : 0.81511
RMSE : 0.90283
```

Item - genre

- **TF-IDF**
  - 장르에 대해 TF-IDF 적용
  - Pearson, Cosine 유사도 측정



title	Horror	Film-Noir	Short	IMAX	Thriller	Western	Crime	Adventure	Romance	Sci-Fi	...	Mystery	Comedy	Animation	Action	War
Sabrina1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.026552	0.0	...	0.0	0.016454	0.0	0.000000	0.000000
Persuasion1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.026552	0.0	...	0.0	0.000000	0.0	0.000000	0.000000
Misérables, Les1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.000000	0.0	0.000000	0.054905
Black Sheep1	0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.000000	0.000000	0.0	...	0.0	0.018592	0.0	0.000000	0.000000
Broken Arrow1	0.0	0.0	0.0	0.0	0.022757	0.0	0.0	0.028956	0.000000	0.0	...	0.0	0.000000	0.0	0.023502	0.000000

Item - genre

- **TF-IDF - Pearson**
  - Pearson 유사도 계산

```
# TF-IDF 변환
transformer = TfidfTransformer()
genre_tfidf = transformer.fit_transform(movie_genre.T).T

# TF-IDF 결과를 데이터프레임으로 변환
genre_tfidf_df = pd.DataFrame(genre_tfidf.toarray(), columns=movie_genre.columns, index=movie_genre.index)

# 결과 출력
genre_tfidf_df
```

이전 대비 성능 감소

```
MAE  : 0.79618
MSE  : 1.11368
RMSE : 1.05531
```

Item - genre

- **TF-IDF - Cosine**
  - Cosine 유사도 계산

```
# TF-IDF 변환
transformer = TfidfTransformer()
genre_tfidf = transformer.fit_transform(movie_genre.T).T

# TF-IDF 결과를 데이터프레임으로 변환
genre_tfidf_df = pd.DataFrame(genre_tfidf.toarray(), columns=movie_genre.columns, index=movie_genre.index)

# 결과 출력
genre_tfidf_df
```

Pearson 대비 성능 향상

```
MAE  : 0.7096
MSE  : 0.81163
RMSE : 0.9009
```

Item - genre

- **Item-genre results**
  - 전체적으로 Cosine 유사도의 지표가 높게 나타남
  - TF-IDF가 가장 좋은 성능

	Baseline Pearson	Baseline Cosine	TF-IDF Pearson	TF-IDF Cosine
MAE	0.7548	0.70819	0.79618	0.7096
MSE	0.94524	0.81511	1.11368	0.81163
RMSE	0.97224	0.90283	1.05531	0.9009

Outline

1. **Introduction**
   - **1.1 Project goal**
   - **1.2 Pipeline**
2. **Data preprocessing**
   - **2.1 Genre missing value**
   - **2.2 Title duplicated**
   - **2.3 Dataset**
3. **Item – genre**
   - **3.1 Baseline**
   - **3.2 TF-IDF**
4. **Item - user**
   - **4.1 Baseline**
   - **4.2 KNN**
   - **4.3 SVD**
5. **Result**
   - **5.1 How to use**
   - **5.2 Limitation**

Item - user

- **Baseline**
  - Rating 점수 결측 0
  - Pearson, Cosine 유사도 측정

userId	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
title																					
'71	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	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	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	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
'Til There Was You	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
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
eXistenZ	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	4.5	0.0	0.0
xXx	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.5	0.0	2.0
xXx: State of the Union	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	1.5
¡Three Amigos!	4.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
À nous la liberté (Freedom for Us)	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

9719 rows × 610 columns

Item - user

- **Baseline - Pearson**
  - Pearson 유사도 계산

	userId	title	rating	pred_rating
0	453	South Park: Bigger, Longer and Uncut	5.0	3.951813
1	318	Story of Film: An Odyssey, The	4.0	3.862937
2	448	L.A. Story	3.0	2.925399
3	19	Back to School	3.0	2.644714
4	182	Aliens	4.5	3.530559
...	...	...	...	...
19995	220	National Lampoon's Vacation	4.0	3.958189
19996	288	Dick	3.5	3.131384
19997	313	In the Line of Fire	3.0	3.377299
19998	202	Return of the Pink Panther, The	4.0	3.810838
19999	452	Fast and the Furious, The	4.0	4.468303

20000 rows × 4 columns

```
MAE  : 0.68918
MSE  : 0.77199
RMSE : 0.87863
```

Item - user

- **Baseline - Cosine**
  - Cosine 유사도 계산

	userId	title	rating	pred_rating
0	453	South Park: Bigger, Longer and Uncut	5.0	3.975045
1	318	Story of Film: An Odyssey, The	4.0	3.860701
2	448	L.A. Story	3.0	2.980069
3	19	Back to School	3.0	2.658977
4	182	Aliens	4.5	3.530183
...	...	...	...	...
19995	220	National Lampoon's Vacation	4.0	4.002504
19996	288	Dick	3.5	3.149933
19997	313	In the Line of Fire	3.0	3.416253
19998	202	Return of the Pink Panther, The	4.0	3.833738
19999	452	Fast and the Furious, The	4.0	4.493864

20000 rows × 4 columns

Pearson 대비 성능 감소

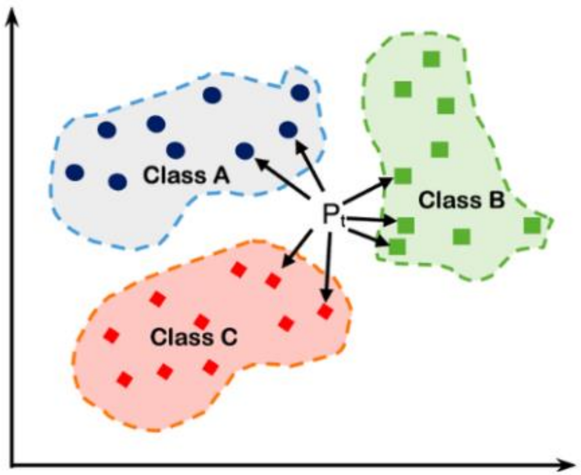
```
MAE  : 0.69465
MSE  : 0.7918
RMSE : 0.88983
```

KWANGWOON UNIVERSITY

Item - user

- **KNN**
  - Rating 점수 결측 KNN 으로 결측 처리
  - Pearson, Cosine 유사도 측정
  - K = 1, 3, 5에 대해 각각 비교 실험 진행

K Nearest Neighbors



title	1	2	3	4	5	6	7	8	9	10	...	601	602	603	604	605	606	607	608	609	610
'71	4.0	2.500000	0.5	5.0	4.0	3.0	0.5	4.0	5.0	3.0	...	4.5	2.0	1.0	4.0	3.0	4.5	4.0	5.0	3.00000	4.0
'Hellboy': The Seeds of Creation	4.0	4.500000	0.5	5.0	3.0	3.0	4.0	3.0	5.0	2.5	...	5.0	5.0	4.0	3.0	3.5	3.5	3.0	4.5	4.00000	5.0
'Round Midnight	4.0	4.000000	0.5	4.0	4.0	5.0	4.5	3.0	4.0	3.0	...	4.0	5.0	1.0	3.0	3.0	3.5	3.0	5.0	3.00000	4.0
'Salem's Lot	4.0	3.948276	0.5	5.0	3.0	4.0	4.0	3.0	5.0	3.5	...	5.0	2.0	5.0	2.0	2.0	4.0	3.0	4.0	3.27027	4.5
'Til There Was You	4.0	3.948276	0.5	5.0	3.0	3.0	2.0	3.0	5.0	3.5	...	5.0	5.0	5.0	3.0	3.0	4.0	3.0	4.0	4.00000	4.5
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
eXistenZ	5.0	2.500000	0.5	4.0	4.0	4.0	3.5	3.0	3.0	0.5	...	4.5	3.0	5.0	4.0	3.5	4.0	3.0	4.5	3.00000	4.0
xXx	4.0	5.000000	4.5	2.0	3.0	2.0	1.0	3.0	1.0	1.5	...	4.0	4.0	3.0	3.0	3.0	4.0	3.0	3.5	3.00000	2.0
xXx: State of the Union	3.0	5.000000	0.5	2.0	3.0	3.0	0.5	3.0	1.0	4.5	...	3.5	3.0	5.0	4.0	3.0	1.0	4.0	3.0	3.00000	1.5
¡Three Amigos!	4.0	2.000000	4.5	3.0	3.0	3.0	4.5	3.0	3.0	3.0	...	4.5	3.0	4.0	3.0	4.0	4.0	3.0	4.5	3.00000	1.0
À nous la liberté (Freedom for Us)	5.0	3.948276	0.5	5.0	5.0	3.0	3.0	5.0	5.0	0.5	...	5.0	5.0	3.0	3.0	4.0	4.0	5.0	4.0	3.00000	3.5

9719 rows × 610 columns

https://medium.com/@sachinsoni600517/k-nearest-neighbours-introduction-to-machine-learning-algorithms-9dbc9d9fb3b2

Item - user

- **KNN - 1**
  - K = 1, Pearson, Cosine 유사도 측정

```python
from sklearn.impute import KNNImputer

# KNNImputer 객체 생성
imputer = KNNImputer(n_neighbors=1)

# pandas dataframe을 numpy array로 변환
matrix = movie_user.to_numpy()

# KNNImputer를 이용하여 결측치를 채움
matrix_imputed = imputer.fit_transform(matrix)

# numpy array를 다시 pandas dataframe으로 변환
movie_user_imputed = pd.DataFrame(matrix_imputed, index=movie_user.index, columns=movie_user.columns)
```

Pearson

```
MAE  : 0.69627
MSE  : 0.78939
RMSE : 0.88848
```

Cosine

```
MAE  : 0.72991
MSE  : 0.87398
RMSE : 0.93487
```

Item - user

- **KNN - 3**
  - K = 3, Pearson, Cosine 유사도 측정

```
# KNNImputer 객체 생성
imputer = KNNImputer(n_neighbors=3)

# pandas dataframe을 numpy array로 변환
matrix = movie_user.to_numpy()

# KNNImputer를 이용하여 결측치를 채움
matrix_imputed = imputer.fit_transform(matrix)

# numpy array를 다시 pandas dataframe으로 변환
movie_user_imputed = pd.DataFrame(matrix_imputed, index=movie_user.index, columns=movie_user.columns)
```

K=1 대비 성능 감소

Pearson

```
MAE  : 0.70055
MSE  : 0.80323
RMSE : 0.89623
```

Cosine

```
MAE  : 0.73027
MSE  : 0.87472
RMSE : 0.93527
```

Item - user

- **KNN - 5**
  - K = 5, Pearson, Cosine 유사도 측정

```
# KNNImputer 객체 생성
imputer = KNNImputer(n_neighbors=5)

# pandas dataframe을 numpy array로 변환
matrix = movie_user.to_numpy()

# KNNImputer를 이용하여 결측치를 채움
matrix_imputed = imputer.fit_transform(matrix)

# numpy array를 다시 pandas dataframe으로 변환
movie_user_imputed = pd.DataFrame(matrix_imputed, index=movie_user.index, columns=movie_user.columns)
```

K=1, 3 대비 성능 감소

Pearson

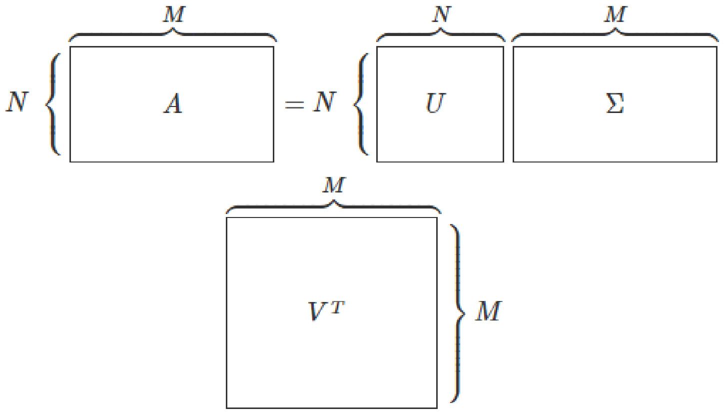
```
MAE  : 0.70398
MSE  : 0.81116
RMSE : 0.90065
```

Cosine

```
MAE  : 0.73038
MSE  : 0.87495
RMSE : 0.93539
```

Item - user

- **SVD**
  - Rating 점수 결측 각각의 컬럼 평균 대체
  - 이후, 각각의 행을 빼줌
  - Pearson, Cosine 유사도 측정
  - N_component 10, 25, 50% 조정 후 측정
  - Iter = 5 설정
  - 잠재요인 기반 CF



userId title	1	2	3	4	5	6	7	8	9	10	...	601	602	603
'71	0.003245	0.001309	-0.005696	-0.000508	-0.000161	-0.000795	-0.001934	-0.000436	-0.001845	-0.001841	...	0.003520	-0.001255	-0.000546
'Hellboy': The Seeds of Creation	0.003276	0.001320	-0.005653	-0.000481	-0.000109	-0.000779	-0.001922	-0.000387	-0.001830	-0.001785	...	0.003527	-0.001212	-0.000576
'Round Midnight	0.003263	0.001328	-0.005619	-0.000504	-0.000106	-0.000766	-0.001962	-0.000386	-0.001821	-0.001755	...	0.003528	-0.001225	-0.000675
'Salem's Lot	0.003255	0.001349	-0.005586	-0.000415	-0.000095	-0.000746	-0.001972	-0.000385	-0.001813	-0.001716	...	0.003531	-0.001232	-0.000647
'Til There Was You	0.003240	0.001343	-0.005617	-0.000406	-0.000084	-0.000736	-0.001905	-0.000374	-0.001829	-0.001763	...	0.003539	-0.001180	-0.000702

https://data-science-hi.tistory.com/82

Item - user

- **Only Normalization**

  - Rating 점수 결측 각각의 컬럼 평균 대체
  - 이후, 각각의 행을 빼줌
  - Cosine 유사도 측정

```python
# 결측을 컬럼 평균으로 채워주는 함수
def R_filled_in(rating_table):
    for col in range(len(rating_table.columns)):
        col_update=[]
        # 컬럼의 평균을 구한다.
        col_num = [i for i in rating_table.iloc[:,col] if math.isnan(i)==False]
        col_mean = sum(col_num)/len(col_num)

        # NaN을 가진 행은 위에서 구한 평균 값으로 채워준다.
        col_update = [i if math.isnan(i)==False else col_mean for i in rating_table.iloc[:,col]]

        # 리스트로 만든 업데이트된 한 컬럼을 기존에 데이터 프레임 컬럼에 새로 입혀준다.
        rating_table.iloc[:,col] = col_update

    return rating_table
```

```
MAE  : 0.72477
MSE  : 0.85644
RMSE : 0.92544
```

Item - user

- **SVD - 60**
  - N_ component = 60

```
# SVD 정의
svd = TruncatedSVD(n_components=60, n_iter=5)

svd.fit(np.array(rating_R_norm))

# 특이값 분해 요소 정의
U=svd.fit_transform(np.array(rating_R_norm))
Sigma=svd.explained_variance_ratio_
VT= svd.components_

# 시그마 행렬에 요소들에 루트
Sigma_sqrt=[]
for i in range(len(Sigma)):
    tmp=[]
    tmp=[math.sqrt(s) for s in np.diag(Sigma)[i]]
    Sigma_sqrt.append(tmp)

Sigma_sqrt=np.array(Sigma_sqrt)

# 특이값분해
ratings_reduced= pd.DataFrame(np.matmul(np.matmul(U, np.diag(Sigma)), VT), columns=movie_user.columns, index=movie_user.index)
```

이전 대비 성능 향상

```
MAE  : 0.66377
MSE  : 0.77411
RMSE : 0.87983
```

Item - user

- **SVD - 150**
  - N_ component = 150

```
# SVD 정의
svd = TruncatedSVD(n_components=150, n_iter=5)

svd.fit(np.array(rating_R_norm))

# 특이값 분해 요소 정의
U=svd.fit_transform(np.array(rating_R_norm))
Sigma=svd.explained_variance_ratio_
VT= svd.components_

# 시그마 행렬에 요소들에 루트
Sigma_sqrt=[]
for i in range(len(Sigma)):
    tmp=[]
    tmp=[math.sqrt(s) for s in np.diag(Sigma)[i]]
    Sigma_sqrt.append(tmp)

Sigma_sqrt=np.array(Sigma_sqrt)

# 특이값분해
ratings_reduced= pd.DataFrame(np.matmul(np.matmul(U, np.diag(Sigma)), VT), columns=movie_user.columns, index=movie_user.index)
```

이전 대비 성능 비슷

```
MAE  : 0.66346
MSE  : 0.77341
RMSE : 0.87943
```

Item - user

- **SVD - 300**
  - N_ component = 300

```python
# SVD 정의
svd = TruncatedSVD(n_components=300, n_iter=5)

svd.fit(np.array(rating_R_norm))

# 특이값 분해 요소 정의
U=svd.fit_transform(np.array(rating_R_norm))
Sigma=svd.explained_variance_ratio_
VT= svd.components_

# 시그마 행렬에 요소들에 루트
Sigma_sqrt=[]
for i in range(len(Sigma)):
    tmp=[]
    tmp=[math.sqrt(s) for s in np.diag(Sigma)[i]]
    Sigma_sqrt.append(tmp)

Sigma_sqrt=np.array(Sigma_sqrt)

# 특이값분해
ratings_reduced= pd.DataFrame(np.matmul(np.matmul(U, np.diag(Sigma)), VT), columns=movie_user.columns, index=movie_user.index)
```

이전 대비 성능 비슷

```
MAE  : 0.66344
MSE  : 0.77336
RMSE : 0.87941
```

Item - user

- **Item-user results**
  - Baseline, SVD가 전반적으로 가장 좋은 지표
  - Baseline Pearson이 가장 좋은 성능을 보임
  - Item-genre 대비 우수한 성능

	Baseline Pearson	Baseline Cosine	KNN1 Pearson	KNN1 Cosine	KNN3 Pearson	KNN3 Cosine	KNN5 Pearson	KNN5 Cosine	Normalization	SVD n_comp 60	SVD n_comp 150	SVD n_comp 300
MAE	0.68918	0.69465	0.69627	0.72991	0.70055	0.73027	0.70398	0.73038	0.72477	0.66377	0.66346	0.66344
MSE	0.77199	0.7918	0.78939	0.87398	0.80323	0.87472	0.81116	0.87495	0.85644	0.77411	0.77341	0.77336
RMSE	0.87863	0.88983	0.88848	0.93487	0.89623	0.93527	0.90065	0.93539	0.92544	0.87983	0.87943	0.87941

Outline

1. Introduction
    1.1 Project goal
    1.2 Pipeline
2. Data preprocessing
    2.1 Genre missing value
    2.2 Title duplicated
    2.3 Dataset
3. Item – genre
    3.1 Baseline
    3.2 TF-IDF
4. Item - user
    4.1 Baseline
    4.2 KNN
    4.3 SVD
5. **Result**
    5.1 How to use
    5.2 Limitation

Result

- **How to use**
  - 사용자 영화 추천을 위한 함수 구현
  - 사용자 정보 입력시 추천

```python
def get_recommendations_with_ranking(model_name, user_id):
    # 선호 장르 입력받기
    print("장르 종류 : Film-Noir, Romance, Documentary, Drama, War, Fantasy, Musical, Family, Sci-Fi,\
    IMAX, Mystery, Action, Music, \n \t    Crime, Western, Biography, Adventure, Children, Thriller,\
     History, Comedy, Animation, Short, Horror", end='\n\n')
    preferred_genre = input("선호하는 장르는 무엇인가요? (상관없으면 ENTER):")
    print('', end='\n\n')

    user_mov = df[df['userId'] == user_id]
    user_mov_pred = model_name[model_name['userId'] == user_id]
    user_mov = pd.merge(user_mov, user_mov_pred, on=['userId', 'title'], how='right')
    user_mov = pd.merge(user_mov, movies[['title', 'genres']], on='title', how='left')

    # 장르를 입력했다면 필터링
    if preferred_genre.strip():
      user_mov = user_mov[user_mov['genres_y'].str.contains(preferred_genre)]

    # 사용자가 아직 안 본 영화
    movie_candidate = user_mov[user_mov['movieId'].isnull()]
    movie_candidate = movie_candidate.sort_values(by='pred_rating', ascending=False)[:10]

    # 랭킹 순으로 결과를 나타내기
    ranked_recommendations = movie_candidate[['title']].reset_index(drop=True)

    # 인덱스에 1씩 추가
    ranked_recommendations.index = ranked_recommendations.index + 1

    # title 컬럼명 변경
    ranked_recommendations = ranked_recommendations.rename(columns={'title': '좋아하실 만한 영화들이에요!'})


    # 결과를 가운데 정렬하는 HTML 스타일 적용
    styled_recommendations = (
        ranked_recommendations.style
        .set_properties(**{'text-align': 'center'})
        .set_table_styles([{'selector': 'th', 'props': [('text-align', 'center')]}])
    )

    return styled_recommendations
```

장르 종류 : Film-Noir, Romance, Documentary, Drama, War, Fantasy, Musical, Family, Sci-Fi, IMAX, Mystery, Action, Music,
Crime, Western, Biography, Adventure, Children, Thriller, History, Comedy, Animation, Short, Horror

선호하는 장르는 무엇인가요? (상관없으면 ENTER):Music

	좋아하실 만한 영화들이에요!
1	Nightmare Before Christmas, The
2	Snow White and the Seven Dwarfs
3	Wizard of Oz, The
4	Willy Wonka & the Chocolate Factory
5	Pinocchio1
6	Blues Brothers, The
7	Mary Poppins
8	South Park: Bigger, Longer and Uncut
9	Little Mermaid, The
10	Shrek 2

Result

- **Limitation**
  - 다양한 조합을 가지는 model hyperparameter tuning 한계
  - Pyspark를 사용한 분산처리 어려움
  - Tag data를 활용하지 못함

https://www.google.com/url?sa=i&url=https%3A%2F%2Fblog.sqlauthority.com%2F2010%2F09%2F27%2Fsql-server-keywords-view-definition-must-not-contain-for-indexed-view-limitation-of-the-view
10%2F&psig=AOvVaw1wTR145dlP7QiOdbS6jiGK&ust=1701797059970000&source=images&cd=vfe&opi=89978449&ved=0CBEQjRxqFwoTCNDs3pum9oID
FQAAAAAdAAAAABAE

Q & A

감사합니다 !

Result

```python
from tqdm.notebook import tqdm

def modeling(similarity_matrix, data):
    df_pred_all = pd.DataFrame()
    users = sorted(data['userId'].unique())
    all_titles = similarity_matrix.index
    n_titles = len(all_titles)

    for user in tqdm(users):
        idx = data[data['userId'] == user].index

        # 유사도
        watched_title = data.loc[idx, 'title'].tolist()
        sub_sim_mat = similarity_matrix.loc[watched_title]
        sub_sim_mat = sub_sim_mat.T.to_numpy()
        sim_N = np.sum(sub_sim_mat, axis=1) + 1

        # 평점 예측
        watched_title_y = data.loc[idx, 'rating']
        watched_title_y = np.array(watched_title_y.tolist()).reshape(-1, 1)

        pred_y = np.matmul(sub_sim_mat, watched_title_y).flatten() / sim_N

        user_list = [user] * n_titles
        cur_pred = pd.DataFrame(zip(user_list, all_titles, pred_y),
                                columns=['userId', 'title', 'pred_rating'])

        # 결과 기록
        df_pred_all = pd.concat([df_pred_all, cur_pred], axis=0)
    return df_pred_all
```