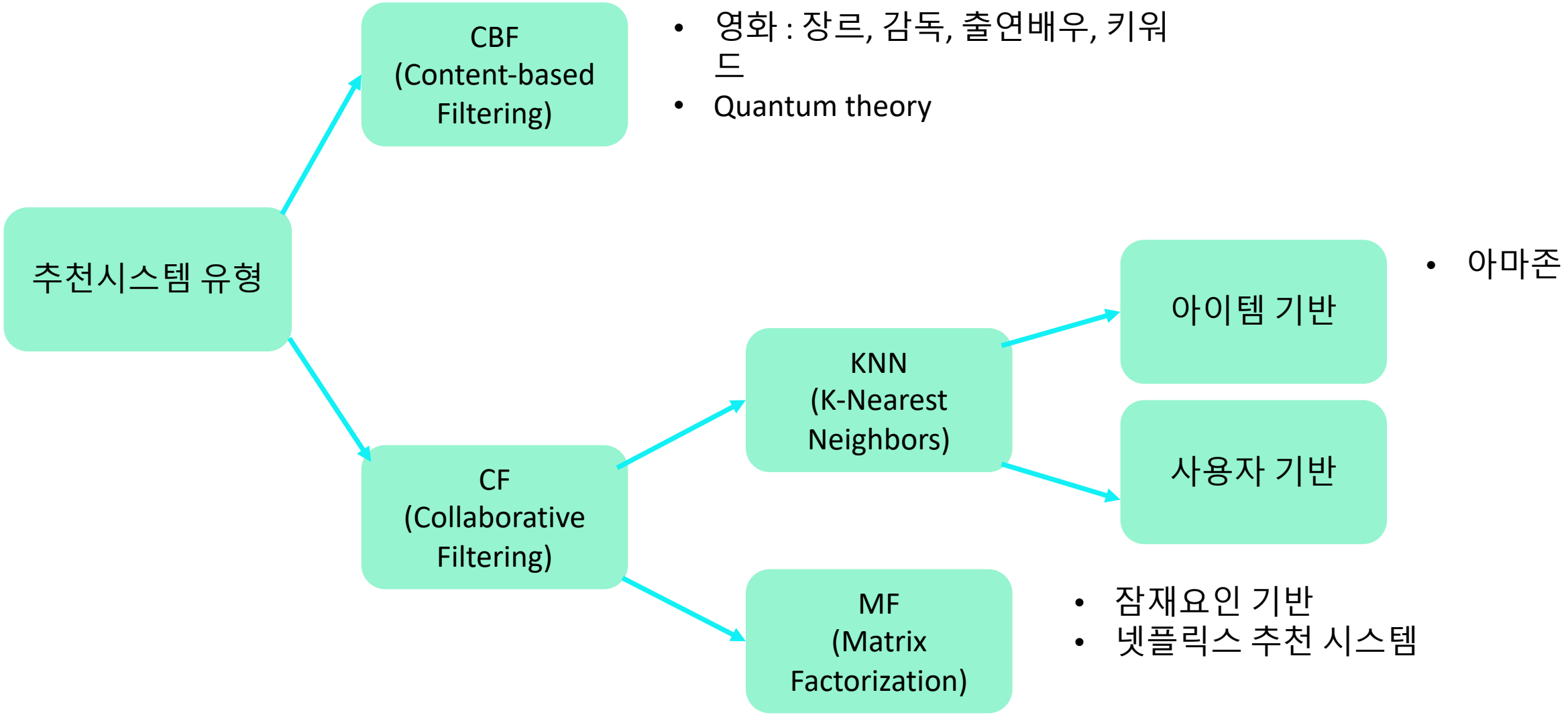
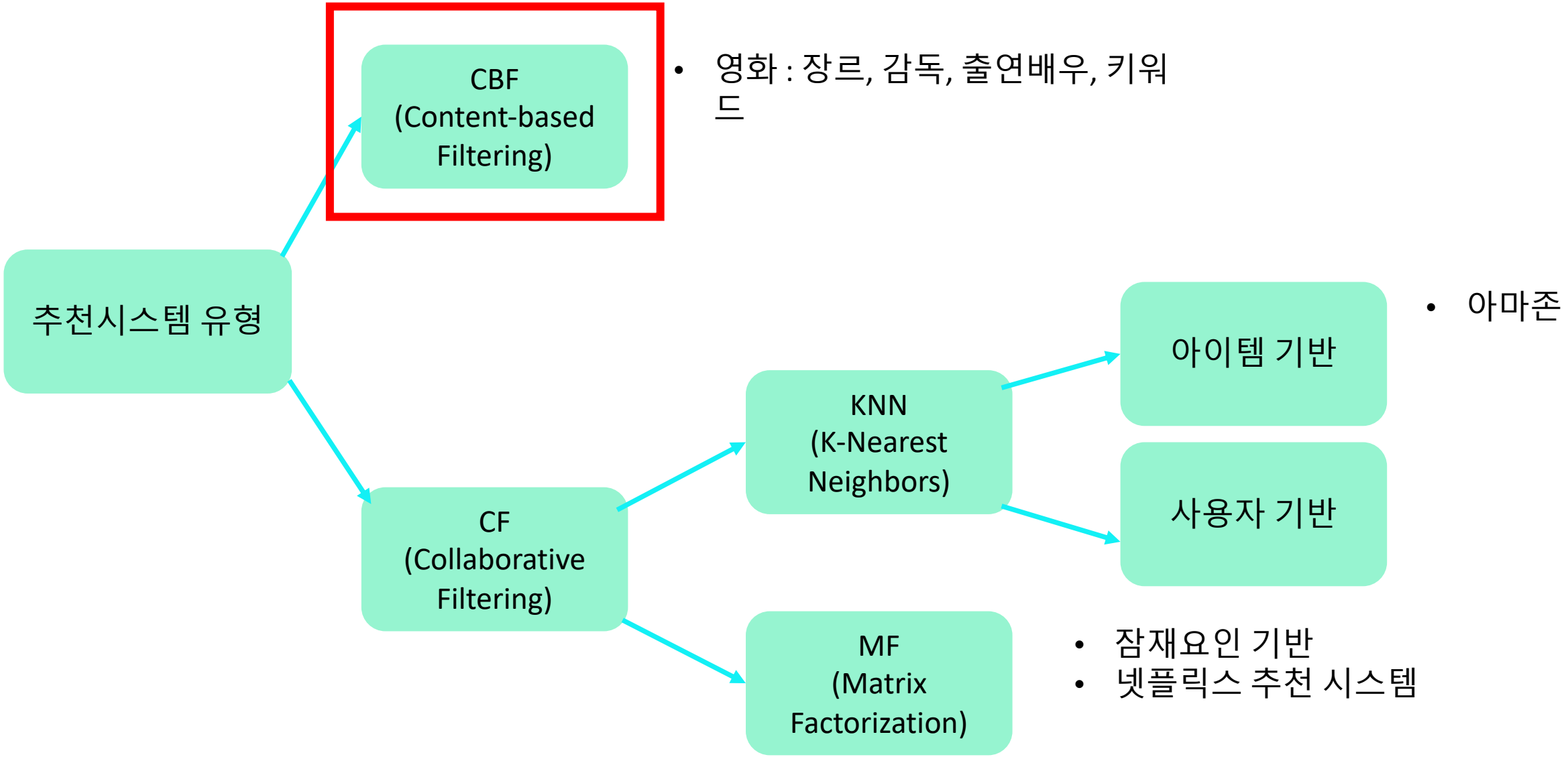
장르 유사도 기반
영화 추천 시스템 (CBF)

추천시스템 종류


추천시스템 유형
CBF
(Content-based
Filtering)
CF
(Collaborative
Filtering)
KNN
(K-Nearest
Neighbors)
MF
(Matrix
Factorization)
아이템 기반
사용자 기반
• 영화 : 장르, 감독, 출연배우, 키워드
• Quantum theory
• 아마존
• 잠재요인 기반
• 넷플릭스 추천 시스템

추천시스템 종류



- 영화 : 장르, 감독, 출연배우, 키워드
- 아마존
- 잠재요인 기반
- 넷플릭스 추천 시스템

영화 데이터 셋

캐글 TMDB 데이터 : https://www.kaggle.com/tmdb/tmdb-movie-metadata

예산	장르	홈페이지	id	키워드	언어	원제목	오버뷰	관람객	제작사	제작국가	개봉일	수익	상영시간	언어	상태	한줄설명	제목	평점	평가참여자수
budget	genres	homepage	id	keywords	origina	original_tit	overview	popularity	productior	productior	release_dat	reven	runtime	spoken_la	status	tagline	title	vote_avera	vote_count
2.37E+08	[{"id": 28, "	http://www	###	[{"id": 146	en	Avatar	In the 22n	150.4376	[{"name": "	[{"iso_316	########	####	162	[{"iso_639_	Released	Enter the \	Avatar	7.2	11800
3E+08	[{"id": 12, "	http://disn	285	[{"id": 270,	en	Pirates of	Captain Ba	139.0826	[{"name": "	[{"iso_316	########	####	169	[{"iso_639_	Released	At the enc	Pirates of the Caribbean: At	6.9	4500
2.45E+08	[{"id": 28, "	http://www	###	[{"id": 470,	en	Spectre	A cryptic r	107.3768	[{"name": "	[{"iso_316	########	####	148	[{"iso_639_	Released	A Plan No	Spectre	6.3	4466
2.5E+08	[{"id": 28, "	http://www	###	[{"id": 849,	en	The Dark K	Following	112.313	[{"name": "	[{"iso_316	########	####	165	[{"iso_639_	Released	The Legen	The Dark Knight Rises	7.6	9106
2.6E+08	[{"id": 28, "	http://mov	###	[{"id": 818,	en	John Carte	John Carte	43.927	[{"name": "	[{"iso_316	########	####	132	[{"iso_639_	Released	Lost in ou	John Carter	6.1	2124
2.58E+08	[{"id": 14, "	http://www	559	[{"id": 851,	en	Spider-Ma	The seemi	115.6998	[{"name": "	[{"iso_316	########	####	139	[{"iso_639_	Released	The battle	Spider-Man 3	5.9	3576
2.6E+08	[{"id": 16, "	http://disn	###	[{"id": 156	en	Tangled	When the	48.68197	[{"name": "	[{"iso_316	########	####	100	[{"iso_639_	Released	They're tal	Tangled	7.4	3330
2.8E+08	[{"id": 28, "	http://mar	###	[{"id": 882	en	Avengers:	When Ton	134.2792	[{"name": "	[{"iso_316	########	####	141	[{"iso_639_	Released	A New Ag	Avengers: Age of Ultron	7.3	6767
2.5E+08	[{"id": 12, "	http://harr	767	[{"id": 616,	en	Harry Pott	As Harry b	98.88564	[{"name": "	[{"iso_316	########	####	153	[{"iso_639_	Released	Dark Secre	Harry Potter and the Half-Blo	7.4	5293

특정 영화에 대해
장르가 유사한 영화를
추천해주는 서비스를
기획해보자

데이터 전처리

In [1]:
```
import pandas as pd
import numpy as np
import warnings; warnings.filterwarnings('ignore')
```

In [2]:
```
movies = pd.read_csv('./data/tmdb_5000_movies.csv')
```

In [3]:
```
print(movies.shape)
movies.head(1)
```

```
(4803, 20)
```

Out[3]:

	budget	genres	homepage	id	keywords	original_language	original_title	overview	popularit
0	237000000	[{"id": 28, "name": "Action"}, {"id": 12, "nam...	http://www.avatarmovie.com/	19995	[{"id": 1463, "name": "culture clash"}, {"id":...	en	Avatar	In the 22nd century, a paraplegic Marine is di...	150.43757

In [5]:
```
movies_df['genres'][0]
```

Out[5]:
```
'[{"id": 28, "name": "Action"}, {"id": 12, "name": "Adventure"}, {"id": 14, "name": "Fantasy"}, {"id": 878, "name": "Science Fiction"}]'
```

In [6]:
```
type(movies_df['genres'][0])
# str 타입인 것을 확인할 수 있다.
```

Out[6]:
```
str
```

genres 칼럼
- str 형태
- 리스트 안에 딕셔너리로 여러 개의 장르 키워드가 저장된 형태 전처리 필요

데이터 전처리

literal_eval : str 형태를 list형태로 바꿔준다.

genres, keywords 칼럼들의 str형태를 list형태로 바꿔주기

In [38]:
```
from ast import literal_eval
movies_df['genres'] = movies_df['genres'].apply(literal_eval)
movies_df['keywords'] = movies_df['keywords'].apply(literal_eval)
```

In [39]:
```
movies_df['genres'][0]
```

Out[39]:
```
[{'id': 28, 'name': 'Action'},
 {'id': 12, 'name': 'Adventure'},
 {'id': 14, 'name': 'Fantasy'},
 {'id': 878, 'name': 'Science Fiction'}]
```

In [40]:
```
type(movies_df['genres'][0])
# str타입에서 list타입으로 바뀐 것을 확인할 수 있다.
```

Out[40]:
```
list
```

데이터 전처리

list 내 여러개 딕셔너리의 name키에 해당하는 값들을 리스트로 변환

In [41]:
```
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])
```

In [45]:
```
movies_df[['genres', 'keywords']][:1]
```

Out[45]:

	genres	keywords
0	[Action, Adventure, Fantasy, Science Fiction]	[culture clash, future, space war, space colon...

genres 안의 딕셔너리들 중 키워드들만 뽑아냄

장르 유사도 기반 영화 추천 시스템

```
@장르 CBF 추천 : 장르를 피처 벡터화한 후 행렬 데이터 값을 코사인 유사도로 계산하기
<프로세스>
1. 장르 피처 벡터화: 문자열로 변환된 genres 칼럼을 Count 기반으로 피처 벡터화 변환
2. 코사인 유사도 계산 : genres 문자열을 피처 벡터화한 행렬로 변환한 데이터 세트를
   코사인 유사도로 비교
3. 평점으로 계산 : 장르 유사도가 높은 영화 중 평점이 높은 순으로 영화 추천
```

CountVectorizer

In [47]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [50]:
```
# 참고 : CountVectorizer에 대하여
# CountVectorizer는 다음 세가지를 수행한다.
# 1. 문서를 토큰 리스트로 변환한다.
# 2. 각 문서에서 토큰의 출현 빈도를 센다.
# 3. 각 문서를 BOW(Bag of Words) 인코딩 벡터로 변환한다.

from sklearn.feature_extraction.text import CountVectorizer
corpus = [
    'This is the first document.',
    'This is the second second document.',
    'And the third one.',
    'Is this the first document?',
    'The last document?',
]
vect = CountVectorizer()
vect.fit(corpus)
vect.vocabulary_
```

Out[50]:
```
{'this': 9,
 'is': 3,
 'the': 7,
 'first': 2,
 'document': 1,
 'second': 6,
 'and': 0,
 'third': 8,
 'one': 5,
 'last': 4}
```

문서를 토큰 리스트로 변환 후,
각 문서에서 토큰의 출현 빈도를 계산
-> BOW 인코딩 벡터로 변환 가능

장르 유사도 기반 영화 추천 시스템

```
@장르 CBF 추천 : 장르를 피처 벡터화한 후 행렬 데이터 값을 코사인 유사도로 계산하기
<프로세스>
1. 장르 피처 벡터화: 문자열로 변환된 genres 칼럼을 Count 기반으로 피처 벡터화 변환
2. 코사인 유사도 계산 : genres 문자열을 피처 벡터화한 행렬로 변환한 데이터 세트를
   코사인 유사도로 비교
3. 평점으로 계산 : 장르 유사도가 높은 영화 중 평점이 높은 순으로 영화 추천
```

장르 유사도 기반 영화 추천 시스템

장르 칼럼의 키워드들을 공백문자 기준으로 join하여 최종적으로 장르 매트릭스(4803행, 276열) 생성

In [48]:
```
# CountVectorizer를 적용하기 위해 공백문자로 word 단위가 구분되는 문자열로 변환.
movies_df['genres_literal'] = movies_df['genres'].apply(lambda x : (' ').join(x))
movies_df['genres_literal']
```

Out[48]:
```
0         Action Adventure Fantasy Science Fiction
1                         Adventure Fantasy Action
2                           Action Adventure Crime
3                       Action Crime Drama Thriller
4                  Action Adventure Science Fiction
5                         Fantasy Action Adventure
6                                 Animation Family
7                  Action Adventure Science Fiction
8                         Adventure Fantasy Family
9                         Action Adventure Fantasy
10        Adventure Fantasy Action Science Fiction
11                 Adventure Action Thriller Crime
12                        Adventure Fantasy Action
13                        Action Adventure Western
14        Action Adventure Fantasy Science Fiction
15                        Adventure Family Fantasy
16                Science Fiction Action Adventure
17                        Adventure Action Fantasy
18                   Action Comedy Science Fiction
19                        Action Adventure Fantasy
```

In [54]:
```
# CountVectorizer로 학습시켰더니 4803개 영화에 대한 276개 장르의 '장르 매트릭스'가 생성되었다.
count_vect = CountVectorizer(min_df=0, ngram_range=(1,2)) #min_df: 단어장에 들어갈 최소빈도, ngram_rang
genre_mat = count_vect.fit_transform(movies_df['genres_literal'])
print(genre_mat.shape)
```

```
(4803, 276)
```

장르 유사도 기반 영화 추천 시스템

코사인 유사도(cosine_similarity)이용해서 영화별 유사도 계산

In [66]:
```
# 코사인 유사도에 의해 4803개 영화 각각 유사한 영화들이 계산됨
from sklearn.metrics.pairwise import cosine_similarity
genre_sim = cosine_similarity(genre_mat, genre_mat)
print(genre_sim.shape)
print(genre_sim[:5])
```

```
(4803, 4803)
[[1.         0.59628479 0.4472136  ... 0.         0.         0.        ]
 [0.59628479 1.         0.4        ... 0.         0.         0.        ]
 [0.4472136  0.4        1.         ... 0.         0.         0.        ]
 [0.12598816 0.16903085 0.3380617  ... 0.12598816 0.         0.        ]
 [0.75592895 0.3380617  0.50709255 ... 0.         0.         0.        ]]
```

영화별 장르 유사도가 계산된 매트릭스(4803, 4803) 생성

In [68]:
```
# 자료를 정렬하는 것이 아니라 순서만 알고 싶다면 argsort
# 유사도가 높은 영화를 앞에서부터 순서대로 보여줌
# 0번째 영화의 경우 유사도 순서 : 0번, 3494번, 813번, ..., 2401 순서
genre_sim_sorted_ind = genre_sim.argsort()[:, ::-1] # 전체를 -1칸 간격으로
print(genre_sim_sorted_ind[:1])
```

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

특정 영화와 유사도가 높은 순서대로 인덱스 번호를 보여줌

장르 유사도 기반 영화 추천 시스템

추천 ver1. 장르 코사인 유사도에 의해 영화를 추천하는 함수

In [93]:

```
def find_sim_movie_ver1(df, sorted_ind, title_name, top_n=10):

    # 인자로 입력된 movies_df DataFrame에서 'title' 컬럼이 입력된 title_name 값인 DataFrame추출
    title_movie = df[df['title'] == title_name]

    # title_named을 가진 DataFrame의 index 객체를 ndarray로 반환하고
    # sorted_ind 인자로 입력된 genre_sim_sorted_ind 객체에서 유사도 순으로 top_n 개의 index 추출
    title_index = title_movie.index.values
    similar_indexes = sorted_ind[title_index, :(top_n)]

    # 추출된 top_n index들 출력. top_n index는 2차원 데이터 임.
    # dataframe에서 index로 사용하기 위해서 1차원 array로 변경
    print(similar_indexes)
    # 2차원 데이터를 1차원으로 변환
    similar_indexes = similar_indexes.reshape(-1)

    return df.iloc[similar_indexes]
```

장르 유사도 기반 영화 추천 시스템

영화 Gotfather와 장르가 유사한 영화 10개 추천

In [94]:
```
similar_movies = find_sim_movie_ver1(movies_df, genre_sim_sorted_ind, 'The Godfather',10)
similar_movies[['title', 'vote_average', 'genres', 'vote_count']]
# 문제 ; 평점 기반으로 추천하고자 하는데, vote_count가 낮은 영화는 제외하고 싶음
```

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

Out[94]:

	title	vote_average	genres	vote_count
2731	The Godfather: Part II	8.3	[Drama, Crime]	3338
1243	Mean Streets	7.2	[Drama, Crime]	345
3636	Light Sleeper	5.7	[Drama, Crime]	15
1946	The Bad Lieutenant: Port of Call - New Orleans	6.0	[Drama, Crime]	326
2640	Things to Do in Denver When You're Dead	6.7	[Drama, Crime]	85
4065	Mi America	0.0	[Drama, Crime]	0
1847	GoodFellas	8.2	[Drama, Crime]	3128
4217	Kids	6.8	[Drama, Crime]	279
883	Catch Me If You Can	7.7	[Drama, Crime]	3795
3866	City of God	8.1	[Drama, Crime]	1814

문제
평가횟수가 현저히 적은 영화들이 추천되는 것도 있음
low quality 추천 문제

우리가 전혀 모르는 영화를 추천받는 것은 엉뚱한 추천 결과를 낳을 수 있음

-> 평가횟수를 반영한 추천 시스템이 필요

장르 유사도 기반 영화 추천 시스템

가중평점(평점&평가횟수) 반영한 영화 추천

```
@ 가중평점(Weighted Rating):

  (v/(v+m))*R + (m/(v+m))*C

- v : 영화별 평점을 투표한 횟수(vote_count) ★ 투표횟수가 많은 영화에 가중치 부여
- m : 평점을 부여하기 위한 최소 투표 횟수 -> 여기서는 투표수 상위 60%
- R : 개별 영화에 대한 평균 평점(vote_average)
- C : 전체 영화에 대한 평균 평점(movies_df['vote_average'].mean()
# C, m은 고정값
# v,R은 영화마다 변동값
```

투표 횟수가 많으면
가중치가 붙는다.

최종적으로
1. 장르가 유사한 영화 중
2. 가중평점이 높은 영화가
추천되게 된다.

In [86]:
```
# 상위 60%에 해당하는 vote_count를 최소 투표 횟수인 m으로 지정
C = movies_df['vote_average'].mean()
m = movies_df['vote_count'].quantile(0.6)
```

In [84]:
```
# C: 전체 영화에 대한 평균평점 = 약 6점
# m: 평점을 부여하기 위한 최소 투표 횟수 = 370회(상위 60% 수준)
print('C:',round(C,3), 'm:',round(m,3))
```

```
C: 6.092 m: 370.2
```

장르 유사도 기반 영화 추천 시스템

가중평점을 계산하는 함수

In [85]:
```
def weighted_vote_average(record):
    v = record['vote_count']
    R = record['vote_average']

    return ( (v/(v+m)) * R ) + ( (m/(m+v)) * C )
```

가중평점을 return값으로 돌려준다.

In [87]:
```
# 기존 데이터에 가중평점 칼럼 추가
movies_df['weighted_vote'] = movies_df.apply(weighted_vote_average, axis=1)
```

장르 유사도 기반 영화 추천 시스템

추천 ver2. 먼저 장르 유사성 높은 영화 20개 선정 후, 가중평점순 10개 선정

In [95]:

```python
def find_sim_movie_ver2(df, sorted_ind, title_name, top_n=10):
    title_movie = df[df['title'] == title_name]
    title_index = title_movie.index.values

    # top_n의 2배에 해당하는 장르 유사성이 높은 index 추출
    similar_indexes = sorted_ind[title_index, :(top_n*2)]
    similar_indexes = similar_indexes.reshape(-1)

    # 기준 영화 index는 제외
    similar_indexes = similar_indexes[similar_indexes != title_index]

    # top_n의 2배에 해당하는 후보군에서 weighted_vote 높은 순으로 top_n 만큼 추출
    return df.iloc[similar_indexes].sort_values('weighted_vote', ascending=False)[:top_n]
```

먼저, 장르 유사성 높은 것 20개를 먼저 뽑은 뒤

최종적으로 상위 10개를 sort해서 보여준다

장르 유사도 기반 영화 추천 시스템

영화 Gotfather에 대해 장르 유사성, 가중평점 반영한 추천 영화 10개를 뽑아보자

In [96]:
```
similar_movies = find_sim_movie_ver2(movies_df, genre_sim_sorted_ind, 'The Godfather',10)
similar_movies[['title', 'vote_average', 'weighted_vote', 'genres', 'vote_count']]
```

Out[96]:

	title	vote_average	weighted_vote	genres	vote_count
2731	The Godfather: Part II	8.3	8.079586	[Drama, Crime]	3338
1847	GoodFellas	8.2	7.976937	[Drama, Crime]	3128
3866	City of God	8.1	7.759693	[Drama, Crime]	1814
1663	Once Upon a Time in America	8.2	7.657811	[Drama, Crime]	1069
883	Catch Me If You Can	7.7	7.557097	[Drama, Crime]	3795
281	American Gangster	7.4	7.141396	[Drama, Crime]	1502
4041	This Is England	7.4	6.739664	[Drama, Crime]	363
1149	American Hustle	6.8	6.717525	[Drama, Crime]	2807
1243	Mean Streets	7.2	6.626569	[Drama, Crime]	345
2839	Rounders	6.9	6.530427	[Drama, Crime]	439

평가 횟수가 반영된
고품질의 추천이 적용된 모습

장르 유사도 기반 영화 추천 시스템

요약 : Gotfather를 좋아하는 사람에게 영화 추천해주기

```
Gotfather장르가 Drama, Crime이다.
우선 Drama, Crime 장르 기준으로 상위 20개 영화를 뽑아보고,
그 중 평가횟수를 반영한 가중평점 기준 상위 10개 영화를 뽑아서 추천해준다.
```

장르 유사도 기반 영화 추천 시스템

응용 : Spider-Man 3 좋아하는 사람 기준으로 장르가 유사한 영화를 추천해주자

In [30]:
```
similar_movies = find_sim_movie_ver2(movies_df, genre_sim_sorted_ind, 'Spider-Man 3',10)
similar_movies[['title', 'vote_average', 'weighted_vote', 'genres', 'vote_count']]
```

Out[30]:

	title	vote_average	weighted_vote	genres	vote_count
329	The Lord of the Rings: The Return of the King	8.1	8.011871	[Adventure, Fantasy, Action]	8064
262	The Lord of the Rings: The Fellowship of the Ring	8.0	7.922175	[Adventure, Fantasy, Action]	8705
330	The Lord of the Rings: The Two Towers	8.0	7.910111	[Adventure, Fantasy, Action]	7487
19	The Hobbit: The Battle of the Five Armies	7.1	7.027274	[Action, Adventure, Fantasy]	4760
98	The Hobbit: An Unexpected Journey	7.0	6.961224	[Adventure, Fantasy, Action]	8297
126	Thor: The Dark World	6.8	6.748873	[Action, Adventure, Fantasy]	4755
30	Spider-Man 2	6.7	6.652034	[Action, Adventure, Fantasy]	4321
129	Thor	6.6	6.572735	[Adventure, Fantasy, Action]	6525
20	The Amazing Spider-Man	6.5	6.478296	[Action, Adventure, Fantasy]	6586
38	The Amazing Spider-Man 2	6.5	6.466812	[Action, Adventure, Fantasy]	4179

장르 유사도 기반 영화 추천 시스템

응용 : Enemy at the Gates 좋아하는 사람 기준으로 장르가 유사한 영화를 추천해주자

In [29]:
```
similar_movies = find_sim_movie_ver2(movies_df, genre_sim_sorted_ind, 'Enemy at the Gates',10)
similar_movies[['title', 'vote_average', 'weighted_vote', 'genres', 'vote_count']]
```

Out[29]:

	title	vote_average	weighted_vote	genres	vote_count
1525	Apocalypse Now	8.0	7.708775	[Drama, War]	2055
2798	The Boy in the Striped Pyjamas	7.7	7.373173	[War, Drama]	1451
2536	The Deer Hunter	7.8	7.310348	[Drama, War]	921
1662	Glory	7.4	6.757199	[War]	383
585	War Horse	7.0	6.753283	[Drama, War]	992
2662	Sarah's Key	7.2	6.475666	[Drama, War]	196
2016	The Water Diviner	6.8	6.472943	[War, Drama]	431
557	Jarhead	6.6	6.434392	[Drama, War]	765
2671	Born on the Fourth of July	6.7	6.405936	[Drama, War]	395
3310	Far from Men	6.6	6.145019	[Drama, War]	43