Recommendation system collaborative Filtering dengan jaccard similairty ¶

Aviel Leonardo Wijaya - 2201766173
Chandra Tan - 2201762931
Kevin Orlando Sutanto - 2201760945
Muhamad Daffa Mennawi - 2201810943
Devinca Limto - 2201758000

Introduction

collaborative filtering merupakan salah satu algorithma digunakan untuk menyusun rekomendasi system. Ada beberapa macam metode digunain, yaitu misalnya Cosine simiilarity, Pearlson Correlation, dan Jaccard Similairity

Metode

Metode yang digunakan adalah Jaccard Similarity.

In [124]:
```

import pandas as pd
import numpy as np
import re
from sklearn.metrics.pairwise import pairwise_distances
```

In [79]:
```
# Read csv
movies = pd.read_csv('./data/movies.csv')
ratings = pd.read_csv('./data/ratings.csv')
```

In [80]:
```
# Get duplicated movies
duplicated_movies = movies.groupby('title').filter(lambda x: len(x) > 1)
```

In [81]:
```
# remove duplicated movies
movies = movies.loc[~movies['movieId'].isin(duplicated_movies["movieId"].values)]
ratings = ratings.loc[~ratings['movieId'].isin(duplicated_movies["movieId"].value
```

In [82]:
```
# get list genres
list_genres = list(set('|'.join(list(movies["genres"].unique())).split('|')))
list_genres.remove('(no genres listed)') # Not used
list_genres
```

Out[82]:
```
['Sci-Fi',
 'Romance',
 'Action',
 'Children',
 'IMAX',
 'Fantasy',
 'Thriller',
 'Western',
 'Horror',
 'Musical',
 'Adventure',
 'Mystery',
 'Documentary',
 'Comedy',
 'War',
 'Film-Noir',
 'Animation',
 'Crime',
 'Drama']
```

In [83]:
```
# if the movie genre is in the list of genre, set it to 1
for genre in list_genres:
    movies[genre] = movies['genres'].map(lambda x: 1 if genre in x else 0)
```

```
In [84]: movies
```

```
Out[84]:
```

	movieId	title	genres	Sci-Fi	Romance	Action	Childr
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy	0	0	0	
1	2	Jumanji (1995)	Adventure|Children|Fantasy	0	0	0	
2	3	Grumpier Old Men (1995)	Comedy|Romance	0	1	0	
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance	0	1	0	
4	5	Father of the Bride Part II (1995)	Comedy	0	0	0	
...	...	...	...	...	...	...	
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy	0	0	1	
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy	0	0	0	
9739	193585	Flint (2017)	Drama	0	0	0	
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation	0	0	1	
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy	0	0	0	

9732 rows × 22 columns

In [85]:
```
movies = movies.drop('genres', axis=1)
```

In [86]:
```
ratings = ratings.drop("timestamp", axis=1)
```

In [87]:
```
movies
```

Out[87]:

	movieId	title	Sci-Fi	Romance	Action	Children	IMAX	Fantasy	Thriller	Western	...	
0	1	Toy Story (1995)	0	0	0	1	0	1	0	0	...	
1	2	Jumanji (1995)	0	0	0	1	0	1	0	0	...	
2	3	Grumpier Old Men (1995)	0	1	0	0	0	0	0	0	...	
3	4	Waiting to Exhale (1995)	0	1	0	0	0	0	0	0	...	
4	5	Father of the Bride Part II (1995)	0	0	0	0	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
9737	193581	Black Butler: Book of the Atlantic (2017)	0	0	1	0	0	1	0	0	...	
9738	193583	No Game No Life: Zero (2017)	0	0	0	0	0	1	0	0	...	
9739	193585	Flint (2017)	0	0	0	0	0	0	0	0	...	
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	0	0	1	0	0	0	0	0	...	
9741	193609	Andrew Dice Clay: Dice Rules (1991)	0	0	0	0	0	0	0	0	...	

9732 rows × 21 columns

In [88]:
```
data = pd.merge(ratings, movies, on='movieId') # join ratings and movies by movie
```

In [89]:
```
data
```

Out[89]:

	userId	movieId	rating	title	Sci-Fi	Romance	Action	Children	IMAX	Fantasy	...
0	1	1	4.0	Toy Story (1995)	0	0	0	1	0	1	...
1	5	1	4.0	Toy Story (1995)	0	0	0	1	0	1	...
2	7	1	4.5	Toy Story (1995)	0	0	0	1	0	1	...
3	15	1	2.5	Toy Story (1995)	0	0	0	1	0	1	...
4	17	1	4.5	Toy Story (1995)	0	0	0	1	0	1	...
...	...	...	...	...	...	...	...	...	...	...	...
100725	610	160341	2.5	Bloodmoon (1997)	0	0	1	0	0	0	...
100726	610	160527	4.5	Sympathy for the Underdog (1971)	0	0	1	0	0	0	...
100727	610	160836	3.0	Hazard (2005)	0	0	1	0	0	0	...
100728	610	163937	3.5	Blair Witch (2016)	0	0	0	0	0	0	...
100729	610	163981	3.5	31 (2016)	0	0	0	0	0	0	...

100730 rows × 23 columns

Mencari Rekomendasi berdasarkan Judul Film

In [96]:
```
def based_name(data, title):
    data = pd.pivot_table(data, index='title', columns=['userId'], values='rating
    data = data.fillna(0)
    jac_sim = 1 - pairwise_distances(data, metric = "hamming") # This is the form
    jac_sim_df = pd.DataFrame(jac_sim, columns=data.index, index=data.index)
#     display(jac_sim_df)
    jac_sim_df = pd.DataFrame(jac_sim_df[title].sort_values(ascending=False))
    jac_sim_df.reset_index(level=0, inplace=True)
    jac_sim_df.columns = ['title','jaccard similarity']
    return jac_sim_df
```

hasil rekomendasi berdasarkan Judul Film

In [97]:
```
title = "No Game No Life: Zero (2017)"
result_based_by_name = based_name(data, title)
print("==================================================================")
print("Based By Title {}".format(title))
print("==================================================================")
result_based_by_name[0:11]
```

```
************************************
Based By Title No Game No Life: Zero (2017)
************************************
```

Out[97]:

	title	jaccard similarity
0	Gintama: The Final Chapter - Be Forever Yorozu...	1.000000
1	Jon Stewart Has Left the Building (2015)	1.000000
2	No Game No Life: Zero (2017)	1.000000
3	Kingsglaive: Final Fantasy XV (2016)	1.000000
4	Gintama: The Movie (2010)	1.000000
5	Shot Caller (2017)	1.000000
6	Flint (2017)	1.000000
7	Bungo Stray Dogs: Dead Apple (2018)	1.000000
8	The Night Is Short, Walk on Girl (2017)	0.998361
9	Steins;Gate the Movie: The Burden of Déjà vu (...	0.998361
10	Black Butler: Book of the Atlantic (2017)	0.998361

Mencari Berdasarkan nama dan genre dari nama film

In [121]:
```
def based_name_genre(jac_sim_df,movies_df,categories):
    name_genre_df = jac_sim_df.merge(movies, on='title')
    columns = ['title', 'jaccard similarity', 'genre similarity']
    name_genre_df['genre similarity'] = [pair_d_row(name_genre_df, 0, row,categor
    df = name_genre_df[columns]\
.sort_values('jaccard similarity',ascending=False)[10:]\
.sort_values('genre similarity', ascending=False)[:10]
    return df

# temp=0
def pair_d_row(dataframe, row_1, row_2, names):
#     global temp
    matrix_row1 = [[dataframe.loc[row_1,name] for name in names]]
    matrix_row2 = [[dataframe.loc[row_2,name] for name in names]]
    result = round((1 - pairwise_distances(matrix_row1, matrix_row2, metric = "ha
#     if(temp == 0) :
#         print(result)
#         temp=1
    return result
```

Hasil Berdasarkan Nama dan Genre Film

In [122]:
```
result_based_by_name_and_genre = based_name_genre(based_name(data, title), movies

print("===================================================================")
print("Based By Title and Genre {}".format(title))
print("===================================================================")
result_based_by_name_and_genre[0:11]
```

```
===================================================================
Based By Title and Genre No Game No Life: Zero (2017)
===================================================================
```

Out[122]:

	title	jaccard similarity	genre similarity
966	Mezzo Forte (1998)	0.996721	1.00000
6917	The Lego Batman Movie (2017)	0.986885	1.00000
9100	Big Hero 6 (2014)	0.931148	1.00000
8190	Beverly Hills Ninja (1997)	0.972131	0.94737
755	Old Lady and the Pigeons, The (La vieille dame...	0.996721	0.94737
8422	Red (2010)	0.965574	0.94737
2187	Game Over, Man! (2018)	0.996721	0.94737
5270	Powerpuff Girls, The (2002)	0.993443	0.94737
1329	Bobik Visiting Barbos (1977)	0.996721	0.94737
5258	D.E.B.S. (2004)	0.993443	0.94737

Discussion and Conclusion

Setelah kami menggunakan Jaccard Similarity untuk membuat Sistem Rekomendasi, dan mencari hasil rekomendasi, Dikatakan hasil yang diperoleh udah sangat akurat. DiKarenakan Dari judul dan genre kami ubah kedalam bentuk vector, sehingga dapat gunakan Jaccard Similarity itu sebuah Collaborative Filtering untuk mendapatkan rekomendasi film yang sesuai. Dan Nyatanya hasilnya itu benar-benar sangat sesuai.

References:

- https://stackoverflow.com/questions/37003272/how-to-compute-jaccard-similarity-from-a-pandas-dataframe (https://stackoverflow.com/questions/37003272/how-to-compute-jaccard-similarity-from-a-pandas-dataframe)
- https://en.wikipedia.org/wiki/Collaborative_filtering (https://en.wikipedia.org/wiki/Collaborative_filtering)
- https://iopscience.iop.org/article/10.1088/1742-6596/1362/1/012130/pdf (https://iopscience.iop.org/article/10.1088/1742-6596/1362/1/012130/pdf)
- https://github.com/berkurka/MovieRecommender/tree/master/data (https://github.com/berkurka/MovieRecommender/tree/master/data)

```
In [ ]:
```