Project : Recommder System for Movies

Recommender System ¶

The objective of a Recommender System is to recommend relevant items for users, based on their preference. Recommender system is prevalent in the digital space. For example, when you go shopping on Amazon, you will notice that Amazon is recommending products on the front page before you even type anything in the search box. Similarly, when you go on YouTube, the top bar of Youtube is typically "videos recommended to you." All these features are based on recommmender system.

What item to recommend to which user is arguably the most important business decision in many digital platforms. For instance, YouTube cannot control which videos that users upload to it. It cannot control which videos users like to watch. Moreoveor, since watching videos is free, YouTube cannot change the price of its items. It does not have inventory either since each video can be viewed as many times as possible. In this case, what could YouTube control? Or in other words, what differentiates a good video streaming service from a bad one? The answer is recommender system.

Types of Recommender Systems

There are **three** types of recommender system. **In this bonus project, we will implement the first one.**

Popularity-based Recommendation

The most obvious system is popularity-based recommendation. In this case, this model recommends to a user the most popular items that the user has not previously consumed. In the movie setting, we will recommend the movie that most users have liked and consumed. In other words, this system utilizes the "widom of the crowds." It usually provides good recommendations for most of the people. Since it is easy to implement, people normally use popularity-based recommendation as a baseline. *Note: this system is not personalized. If both consumers did not watch Movie A and Movie A is the most popular one, both of them will be recommended Movie A.*

Content-based Recommendation

This recommender system leverages the data of one customer's historical actions. This recommender systems first utilizes a set of features to describe an item (for example, for movies, we can use the movie's director, main actor, main actress, genre, etc. to describe the movie). When a user comes in, the system will recommend the movies that are closest to the movie that the users have consumed and liked before in terms of the features. For instance, if a user likes action move from Nolan the most, this system will recommend another action movie from Nolan that this user has not consumed. *Note: we will not implement this system in this bonus project since it requires knowledge about supervised learning. We will come back to this topic at the end of this semester.*

Collaborative Filtering Recommendation

The last type of recommender system is called collaborative filtering. This approach uses the memory of previous users interactions to compute users similarities based on items they've interacted (user-based approach) or compute items similarities based on the users that have interacted with them (item-based approach).

A typical example of this approach is User Neighbourhood-based CF, in which the top-N similar users (usually computed using Pearson correlation) for a user are selected and used to recommend items those similar users liked, but the current user have not interacted yet.

Read-in the preference file

The first exercise is to read in the movie preference csv file.

It returns two things:

1. A dictionary where the key is username and the value is a vector of (-1, 0, 1) that indicates the users preference across movies (in the order of the csv file).
2. A list of strings that indicate the order of column names.
3. A data frame that contains the csv file.

In [1]:
```
import pandas as pd


def read_in_movie_preference():
    file_location = "./data/movie_preference.csv"
    df = None
    column_names = []
    preference = {}

    f = open(file_location, "r")
    df = pd.read_csv(file_location)

    column_names = list(df.columns[1:])   # assign the movie names to "column_nam

    Values = df.values.tolist()          # assign list of value(0,1,-1) to the va
    for i in Values:
        preference[i[0]] = i[1:]


    return [df, column_names, preference]
```

Compute the ranking of most popular movies

Next task is to take the movie preference dataframe and computes the popular ranking of movies from the most popular to the least popular. You should return a list where each element represents the popularity ranking of the movies. The order of the list should reflect the order of the movie names in the dataframe.

In [2]:
```
def movies_popularity_ranking(df, movie_names):

    point = df.sum(axis=0)[1:]         #sum of the preference value for each movi

    movie_popularity_rank = [0 for x in range(len(point))]

    for i in range(len(point)):        # Iterate each point and compare with the
        (r, s) = (1, 1)
        for j in range(len(point)):
            if j != i and point[j] > point[i]:
                r += 1
        movie_popularity_rank[i] = r



    return movie_popularity_rank
```

Recommendation

Last, this function will take in a user's name, it will return a string representing the name of the top movie that this user has not watched and has best popularity ranking (i.e., lowest ranking number). If the user name does not exit, this function should return an empty string. If the user has watched all movies, this function should return an empty string.

In [3]:
```
def Recommendation(movie_popularity_ranking, preference, movie_names, name):
    recommended_movie = ""
    status = preference[name]          # List of value for given person name
    index = []

    for i in range(len(status)):
        if status[i]==0:                    # Create a list of index of movie that
            index.append(i)
        best = min([movie_popularity_rank[i] for i in index])  # Compare the rank
        recommend_index = movie_popularity_rank.index(best)   # Find the index of

    recommended_movie = movie_names[recommend_index] # Assign the movie names wit




    return recommended_movie
```

In [6]:
```
[df, movie_names, preference] = read_in_movie_preference()
movie_popularity_rank = movies_popularity_ranking(df, movie_names)
df
```

Out[6]:

	Please fill in your name. (You can also use an alias name).	The Shawshank Redemption	The Godfather	The Dark Knight	Star Wars: The Force Awakens	The Lord of the Rings: The Return of the King	Inception	The Matrix	Avengers: Infinity War	Int
0	DJZ	0	1	1	0	1	1	1	-1	
1	Bing He	0	0	0	0	0	0	1	0	
2	Yu	0	0	0	0	0	0	0	1	
3	Kelly Hsieh	1	1	1	1	0	1	1	0	
4	Chris Chen	0	1	1	1	1	0	1	1	
...	...	...	...	...	...	...	...	...	...	
181	Ming Huang	0	-1	-1	-1	-1	1	-1	-1	
182	Lucia Zhang	1	1	0	-1	-1	1	1	0	
183	Jin Hu	1	0	1	0	1	1	0	1	
184	Lihui Jiang	-1	-1	-1	1	1	-1	-1	1	
185	Zhuoqing Bao	1	1	1	1	1	1	1	1	

186 rows × 21 columns

In [7]:
```
Recommendation(movie_popularity_rank, preference, movie_names, "DJZ")
```

Out[7]:
```
'The Shawshank Redemption'
```