In [1]:

```
import pandas as pd
import numpy as np
# 编写UCF推荐算法
def UBCF_rec(train_df,s_data,N):
    rec_pre = pd.DataFrame()
    for u in s_data.index:
        # 获取推荐电影
        try:
            s_users = s_data.loc[u].dropna().sort_values().index[int('-'+str(N)):]
            s_items = []
            for v in s_users:
                s_items += list(train_df.loc[[v]].dropna(axis=1).columns)
            u_items = train_df.loc[u].dropna().index
            rec_items = list(set([i for i in s_items if i not in u_items]))
            # 获取预测评分
            s_uv = s_data.loc[u,list(s_users)].values
            train_df_tmp = train_df.loc[s_users,rec_items].values
            r_m = np.nanmean(train_df.loc[s_users],axis=1)
            U_array = (train_df_tmp-(r_m.reshape(-1,1)*np.ones((N,train_df_tmp.shape[1]))))*s_uv.
            U = np.nansum(U_array,axis=0)
            s_uv_tmp = s_uv.reshape(-1,1)*np.ones((len(s_users),train_df_tmp.shape[1]))*(train_df
            D = np.nansum(s_uv_tmp*r_m.reshape(-1,1),axis=0)
            D = np.nansum(s_uv_tmp,axis=0)
            p = np.mean(r_m) + U/D
            rec_pre_tmp = pd.DataFrame(columns=rec_items)
            rec_pre_tmp.loc[u,:]=p
            rec_pre = pd.concat([rec_pre,rec_pre_tmp])
        except:
            pass
    return rec_pre
```

In [2]:

```
import time
if __name__ == '__main__':
    train_data = pd.read_csv('data/train_data.csv')
    test_data = pd.read_csv('data/test_data.csv')
    # 用户—物品矩阵
    train_df = train_data.pivot(index='user_id', columns='anime_id', values='rating')
    test_df = test_data.pivot(index='user_id', columns='anime_id', values='rating')
    # 基于用户的协同过滤算法# 用户相似度矩阵
    print('请稍等：', end='')
    s_data = pd.DataFrame(index=test_df.index, columns=train_df.index)
    for u in test_df.index:
        Du = np.sqrt(sum(train_df.loc[[u]].dropna(axis=1).values[0] ** 2))
        for v in train_df.index:
            if v != u:
                uv_data = train_df.loc[[u, v]].dropna(axis=1)
                v_data = train_df.loc[[v]].dropna(axis=1)
                U = sum(uv_data.values[0] * uv_data.values[1])
                Dv = np.sqrt(sum(v_data.values[0] ** 2))
                s = U / (Du * Dv)
                s_data.loc[u, v] = s
    print('.', end='')
    # 获取最优近邻数
    print('请稍等：', end='')
    MAE_anchor = []
    time_anchor = []
    for N in range(1, 3, 1):
        start = time.time()
        rec_pre = UBCF_rec(train_df, s_data, N)
        co = list(set(test_df.columns) & set(rec_pre.columns))
        rec_pre1 = rec_pre[co]
        test_df1 = test_df[co]
        MAE = 0
        for u in test_df1.index:
            tmp = pd.concat([rec_pre1.loc[[u]], test_df1.loc[[u]]]).dropna(axis=1)
            MAE += (np.abs(tmp.iloc[0].values - tmp.iloc[1].values)).mean()
        MAE = MAE / rec_pre1.shape[0]
        end = time.time()
        T = end - start
        MAE_anchor.append(MAE)
        time_anchor.append(T)
    print('.', end='')
    # 模型和评价
    rec_pre = UBCF_rec(train_df, s_data, 3)
    co = list(set(test_df.columns) & set(rec_pre.columns))  # 位运算
    rec_pre1 = rec_pre[co]
    test_df1 = test_df[co]
    MAE = []
    for u in test_df1.index:
        tmp = pd.concat([rec_pre1.loc[[u]], test_df1.loc[[u]]]).dropna(axis=1)
        MAE.append(np.abs(tmp.iloc[0].values - tmp.iloc[1].values).mean())
    MAE = np.nanmean(MAE)
    print('预测评分与实际评分之间的均方误差为：', MAE)
    # 写出推荐结果
    rec_pre.to_csv('rec_pre.csv')
```

请稍等：.请稍等：.预测评分与实际评分之间的均方误差为： 1.522342552675616

In [ ]:

```
```