Movie Recommendation with Map-Reduce

R07942041 楊承運

Abstract

In this project, we aim to learn if a movie is worth recommended based on the feedback of audiences. The dataset we are analyzing is MovieLens 1M dataset(https://grouplens.org/datasets/movielens/1m/). We make use of the numbers and scores of feedbacks from audiences, and use Map-Reduce to run on top of Hadoop. The procedure can be divided into three jobs: Pre-processing, Feature induction, and Recommendation. Each job has their own mapper and reducer, which need to run with Hadoop Streaming. To achieve real-time data storage and visualization, we choose MongoDB as the database.

Approach

To accomplish the Recommendation work, there are three steps: Pre-process, Item pairing, Recommendation.

(i) Pre-process

When a list of rating from audience is received, useful information is reserved by pre-processing. Mapper will map every user to every movie they rate first. Then the Reducer will create two kinds of lists by gathering all movies rated by a particular user, and all users who rate a particular movie.

(ii) Item pairing

In this step, Mapper lists all combinations of two movies being rated by the same user. Reducer then scans through all users' ratings to see whether the combination is popular.

(iii) Recommendation

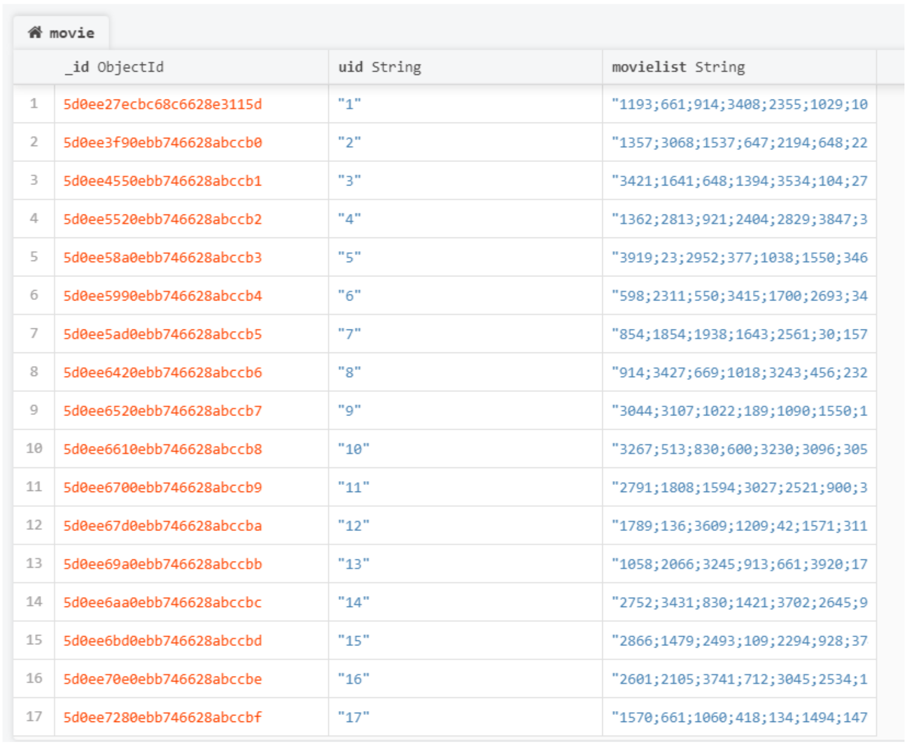
Since the system aims to recommend movies that a user has not seen yet, mapper will list two user lists each containing only one movie of a combination. Later, Reducer add scores of potential combinations from the previous step to the recommendation list of every user.

After the three map-reduce stages, system finally learns how to recommend movies to a user who has seen some movies based on a particular dataset. Take an easy example: if user A has seen movie 1, 2, 3, how to determine which movie, 4 or 5, is more suitable for A? In our method, we review all users' past experiences in the dataset, and calculate the frequency of (1, 4), (2, 4), (3, 4) and (1, 5), (2, 5), (3, 5). If combinations of movies in A with movie 4 appear more than movie 5, then we'll recommend movie 4 to user A.

Results

✧ <u>Pre-processing</u>

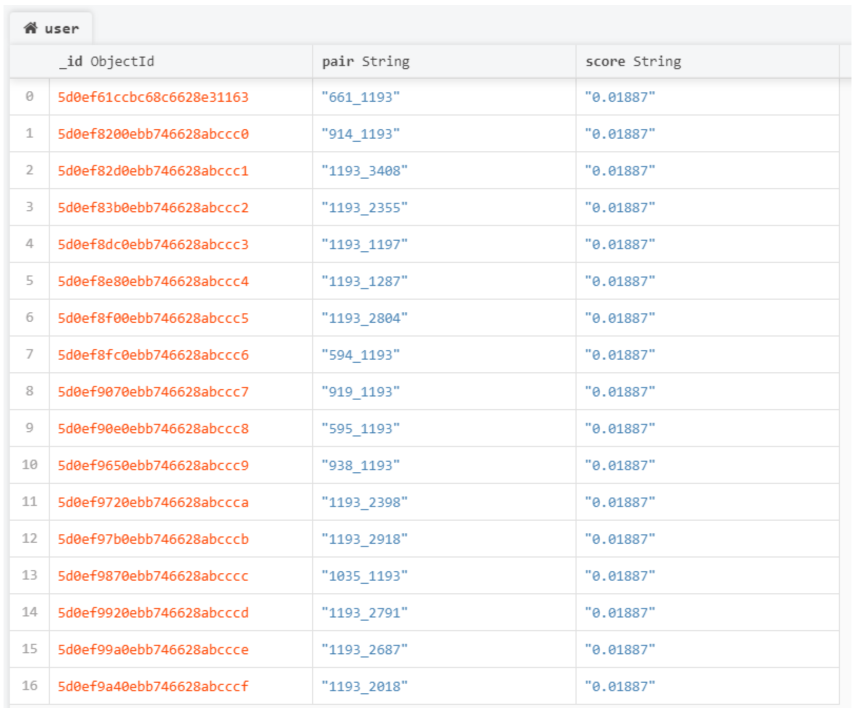
We generate every user's movie list and every movie's user list from the raw data at this stage. All lists are then inserted into MongoDB docker container. Below we use *MongoDB Compass* to connect and show the movie lists in MongoDB after pre-processing.


movie

	_id ObjectId	uid String	movielist String
1	5d0ee27ecbc68c6628e3115d	"1"	"1193;661;914;3408;2355;1029;10
2	5d0ee3f90ebb746628abccb0	"2"	"1357;3068;1537;647;2194;648;22
3	5d0ee4550ebb746628abccb1	"3"	"3421;1641;648;1394;3534;104;27
4	5d0ee5520ebb746628abccb2	"4"	"1362;2813;921;2404;2829;3847;3
5	5d0ee58a0ebb746628abccb3	"5"	"3919;23;2952;377;1038;1550;346
6	5d0ee5990ebb746628abccb4	"6"	"598;2311;550;3415;1700;2693;34
7	5d0ee5ad0ebb746628abccb5	"7"	"854;1854;1938;1643;2561;30;157
8	5d0ee6420ebb746628abccb6	"8"	"914;3427;669;1018;3243;456;232
9	5d0ee6520ebb746628abccb7	"9"	"3044;3107;1022;189;1090;1550;1
10	5d0ee6610ebb746628abccb8	"10"	"3267;513;830;600;3230;3096;305
11	5d0ee6700ebb746628abccb9	"11"	"2791;1808;1594;3027;2521;900;3
12	5d0ee67d0ebb746628abccba	"12"	"1789;136;3609;1209;42;1571;311
13	5d0ee69a0ebb746628abccbb	"13"	"1058;2066;3245;913;661;3920;17
14	5d0ee6aa0ebb746628abccbc	"14"	"2752;3431;830;1421;3702;2645;9
15	5d0ee6bd0ebb746628abccbd	"15"	"2866;1479;2493;109;2294;928;37
16	5d0ee70e0ebb746628abccbe	"16"	"2601;2105;3741;712;3045;2534;1
17	5d0ee7280ebb746628abccbf	"17"	"1570;661;1060;418;134;1494;147

✧ <u>Item-paring</u>

At this stage, we list all possible item pairs and calculate their recommendation scores. Here we simply take the reciprocal of movie numbers watched by one user as the score. That assumes each audience has same recommendation power. Some preview of item pairs is shown below:

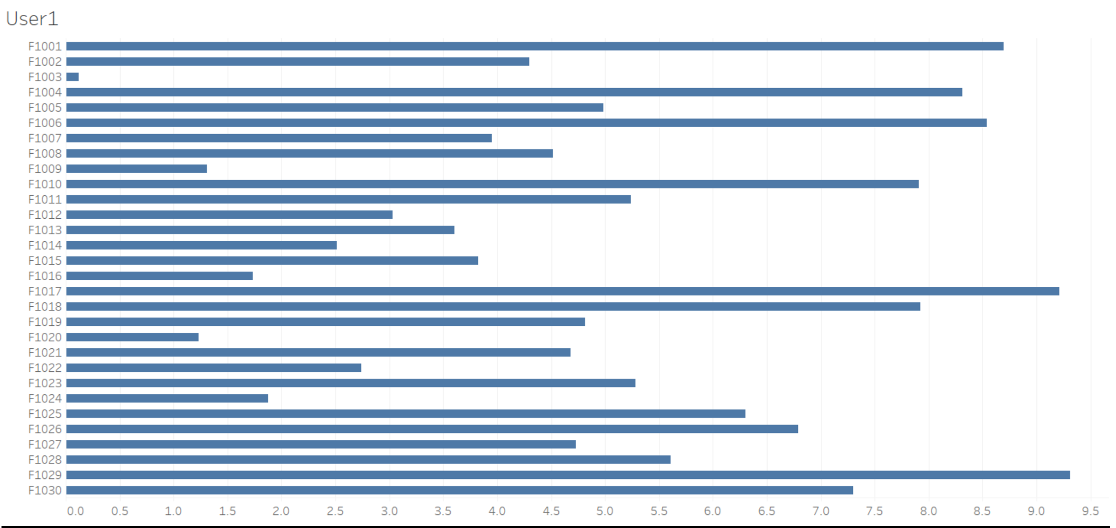

user

	_id ObjectId	pair String	score String
0	5d0ef61ccbc68c6628e31163	"661_1193"	"0.01887"
1	5d0ef8200ebb746628abccc0	"914_1193"	"0.01887"
2	5d0ef82d0ebb746628abccc1	"1193_3408"	"0.01887"
3	5d0ef83b0ebb746628abccc2	"1193_2355"	"0.01887"
4	5d0ef8dc0ebb746628abccc3	"1193_1197"	"0.01887"
5	5d0ef8e80ebb746628abccc4	"1193_1287"	"0.01887"
6	5d0ef8f00ebb746628abccc5	"1193_2804"	"0.01887"
7	5d0ef8fc0ebb746628abccc6	"594_1193"	"0.01887"
8	5d0ef9070ebb746628abccc7	"919_1193"	"0.01887"
9	5d0ef90e0ebb746628abccc8	"595_1193"	"0.01887"
10	5d0ef9650ebb746628abccc9	"938_1193"	"0.01887"
11	5d0ef9720ebb746628abccca	"1193_2398"	"0.01887"
12	5d0ef97b0ebb746628abcccb	"1193_2918"	"0.01887"
13	5d0ef9870ebb746628abcccc	"1035_1193"	"0.01887"
14	5d0ef9920ebb746628abcccd	"1193_2791"	"0.01887"
15	5d0ef99a0ebb746628abccce	"1193_2687"	"0.01887"
16	5d0ef9a40ebb746628abcccf	"1193_2018"	"0.01887"

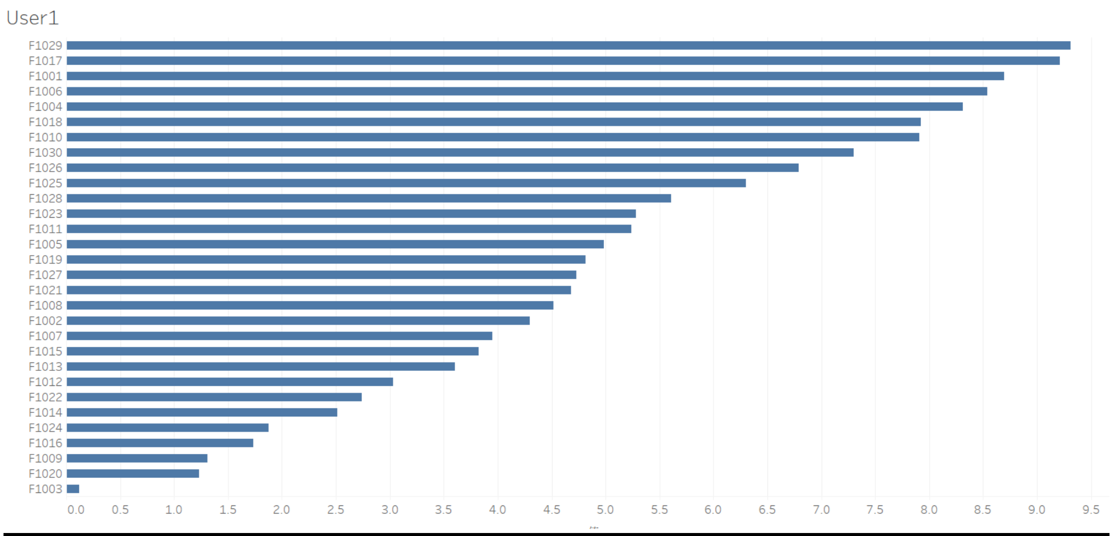
✧ **Recommendation**

At this stage, we find every item's user list from mongoDB and add scores retrieved in the precious step to every user's recommendation items. Here we assume each pair has same recommendation power. After accumulating all scores, the recommendation work is completed.

Following we use *Tableau* to show the recommendation list for user1 (Given 30 different movies):



Reordering the chart by recommendation scores:



We can see the top 5 movies worth recommended to user 1 are no.29, 17, 1, 6, 4.

Discussions

1. **Combine more movies**

In our approach we consider the combination of two movies in one user's movie list as a key factor for recommendation. If bigger movie sets (3 or 4) we consider, the recommendation result may be closer to the user's requirement.

2. **Make use of rating information**

Originally when we generate movie pairs, we assume the user has same preference to each movie. If we introduce the ratings from user, we can add this weight to recommendation score of each movie pair so the result will more coordinate to users' preferences.

3. **Some Default Assumptions**

We have set up some assumptions in our map-reduce procedures, like each user and each item has same recommendation power. Since we don't have more information about users and movies, we set up such assumptions for convenience. If other dataset is chosen or some other particular requirements need to be met, definition of recommendation score can be tuned to fit.

4. Constraints to New Users

The system is unable to recommend movies to new user, due to no movie lists can be related. An alternative is to let the new user select a few movies he is interested in, and recommend following the list.