CMPE 239 Project A: Recommender Systems

Team 2: Xiaoyan Chong, Weiqian Hou, and Fengmei Liu

I. Introduction

In this project, each member of our group tried all the parts independently, and then discussed the solutions together. After that we divided the final implementation into three parts. Weiqian is in charge of Part A, E, and EC2. Fengmei is in charge of Part D, including data preprocessing and analyzing specially for content-based recommendation, and also Part E. Xiaoyan is in charge of Part B, C and EC1. In the report, each people wrote the corresponding parts as divided above. The code reviewing distribution is: Fengmei reviews Weiqian's code, Weiqian reviews Xiaoyan's code, and Xiaoyan reviews Fengmei's Code.

II. Part A: Data Preprocessing

In this part, we would like to get a cleaned and well-structured dataset whose columns are userID, itemID, and value of rating based on Form Responses 1. This dataset would be used in part B, C, EC1, and EC2. Second, we need to process the Movies/Genres information, and output a binary table about genre of movie. Also, we make a label file which give each movie an itemID, such that we can find the movie name for the corresponding itemID.

1. User-Item file

Step 1. Delete the duplicated column AG (Pretty women) in Form Responses 1.

Step 2. Replace N/A with " ".

Step 3. Add the ten new movies into this rating sheet from Movies/Genres table. Let team members rate the new movies first. Then, randomly select some other users and predict their rates based on their "Preferred Genres" and previous rating.

Step 4. We find there are some duplicated userID like 831, 179, 705, 913. So we rename the duplicated one to 83102, 17902, 70502, 91302.

Step 5. Reshape the dataset to what we want. That is, 1st column: userID, 2nd column: itemID, 3rd column: value of rating.

2. Movie-Genre file

Step 1. Delete the duplicated movie, e.g. Star Wars. We keep the first one, because the genre of Star Wars seems not correct for second one.

Step 2. Complete the Movies/Genre table. We google the genres for the movies which missed in the Genre table but appeared in Form Responses 1.

Step 3. Replace "x" with 1 and NA with 0 to make it a binary profile.

3. Label file

Step 1. Extract movie names from the Form Responses 1 and reshape it to a column in sequence.

Step 2. Add a column with number 1 to 37 to it, which is consistent with itemID. Then, output a file with itemID and corresponding movie names.

III. Part B: User-based Collaborative Filtering

In this part, we implement the User-based Collaborative Filtering using Mahout libraries. The steps are shown as follows.

Step 1: Loading data from our data file using the interface called *DataModel*. The dataset is the one we obtained from Part A, which is in the format of userID, itemID and ratings.

Step 2: Create similarity matrix between users by computing the correlation coefficient between them, and we use Pearson Correlation Similarity in this step.

Step 3: Define which similar users we would like to leverage for the recommender. We set the similarity threshold at 0.1 and will use all that have a similarity greater than 0.1.

Step 4: Build a user-based recommender based on all the previous steps.

Step 5: Call the recommender we created, and obtain recommendations for a particular user. We choose to recommend each user top five movies according to the values of rating.

IV. Part C: Item-based Collaborative Filtering

Item-based Collaborative Filtering is also implemented by using Mahout libraries. The steps are show below.

Step 1: Loading data from our data file using the interface called *DataModel*. The dataset is in the format of userID, itemID and ratings.

Step 2: Create similarity matrix between items by computing the correlation coefficient between items, and we use Pearson Correlation Similarity in this step again.

Step 3: Build an item-based recommender based on step 1 and step 2.

Step 4: Call the recommender we created in step 3, and make recommendations for a particular user the top five movies according to the values of rating.

V. Part D: Content-based Recommendations

The content-based recommendation is implemented in R. The general idea is as follows: create Item Profiles by the movie-genre matrix, calculate User Profiles by using both the user-item utility matrix and Item profiles, and recommend new movies to a particular user by comparing the Cosine similarity between the user and all unrated items. Finally, we implement the evaluation of the top 5 recommendations by calculating precision@5. The implementation is shown below.

STEP 0: Install Packages in R

R has the library "RecommenderLab" which is mainly designed for User-based CF and Item-based CF. Here we use it to generate rating matrix to the format of "UserID, ItemID, Rating" and User-Item matrix (row: Users, column: Items). Also it requires installing packages of Matrix, Registry, Arules. We have some text like movie names to handle with, so the package LSA is suggested to install too.

STEP 1: Data Pre-Processing

Because the content-based recommendation can recommend new items, so the new movies listed in the movie genre dataset aren't added in the rating data set. The data processing includes:

User-Item file: 1) Rename the duplicated users, ID179, 705,831,913 to 17902,70502,83102,91302; 2) For the reason of eliminating new user "cold start" problem, delete the low rating users: Id75 (0 ratings) and ID 254 (2 ratings). 3) Rename the duplicated movie Pretty woman to Pretty woman1. 4) Final matrix is User-Item matrix with the format of (row: User, col: Item).

Movie-Genre file: 1) Change the duplicated "Star Wars" to "Star Wars II".

STEP 2: Create Item Profiles

Read the movie genre file, and make it to be the format of (row: Items, column: Genres), with changing the "x" to be 1 and blank to be 0.

STEP 3: Create User Profiles

1) Match the User-Item matrix with the Item Profiles to create a "Matching Item Profile matrix", which is a new Item Profile that the items have the same sequence of the items in User-Item profiles (and if there's no genre information of a movie, the genres will all set to be 0.)

2) *User Profile matrix = User-Item matrix * Matching Item profile matrix*

3) Make User Profile a binary matrix by setting all non-zero values to 1. The format of User Profile is (row: Users, column: Genres)

STEP 4: Calculate Similarity between Users and Items

Calculate Cosine similarity of all users and all items(any movie with genre information) by using:

Similaritymatrix[i,j] = Cosine(UserProfile [i,],ItemProfile[j,])

STEP 5: Recommend to users of new unrated items

1) List user-rated movies, user-unrated movies (including the new movies shown only in Genre file);
2) For the unrated movies, find the similarities from Similarity matrix;
3) Rank the similarities above;

4) Choose Top 5 in the list above

EVALUATION

STEP 1: Split the data to Training data (70%) and Test data(30%); For the Test data, make the lists of Test_Known(each user has 5 rated movies, for recommendations) and Test_Unknown(contains other movies except the 5, for comparison with our prediction later)

STEP 2: Use the Training data to build the content-based recommendation. This time, the User Profile and Similarity matrix calculation would use the Training data and the Test_Known data. Recommend for the users in Test_Unknown matrix.

STEP 3: Get all top-5 recommendations for each of the Test_Unknown users. Match them with the original Test_unknown matrix; Calculate precision@5 by using the formula:
# of correct predictions / # of all ratings to be predicted

RESULT: Precision@5 is 0.63;

VI. Part E: Matrix Factorization

We use Apache Spark Matrix Factor based Recommendations(with ALS method) in this part

Step 1: Prepare two datasets: ratings.csv and movies.csv by using R. As for ratings.csv, we just add a new column timestamp which is random integer 1:100 to the dataset used in part B and C. The ratings.csv includes columns: UserID, MovieID, Rating, and timestamp. The Label file in a (itemID and title) format is the same as in part A.

Step 2: Split data to training(60%), validation(20%) and test(20%).

Step 3: Train models with sparks ALS.train method using training data (60%) and apply it on validation data (20%) to compare accuracy. Adjust the parameters to see which model yields the smallest MSE. We tried 7,10,12 for rank, 5, 6, 7, 10 for number of iteration, and 0.01, 0.05, 0.1, 1 for lambda. It turns out ALS(rank =10, iteration = 6, lambda = 0.1) is the best model with MSE = 1.51.

Step 4: Get recommendations for a particular user.

Step 5: Evaluate the best model on test data (20%) to calculate the precision rate for 5 recommendations and get result of 0.07, and the RMSE is 1.1886.

VII. Extra credit #1 (EC1): Experimental Evaluation

In this section, we evaluated and compared our recommendation engines that we build in our project. The evaluation for each engine (except for content-based recommendation), we use the same dataset which is obtained by Part A. We consider the whole dataset in the evaluation process, and split data into training set (70%) and testing set (30%). We train our recommender using the training set and detect how well it performs on the testing set.

Table 1 summarizes evaluations for all recommenders. Both RMSE and precision@5 are given for each recommender if applicable. Since User-based CF, Item-based CF and Matrix Factorization predict movies according to ratings, RMSE is more accurate for comparing these three recommender. We list the precision just for references when comparing with hybrid recommender. Based on RMSE, we conclude User-based CF performs best among the three. Comparing hybrid recommender with User-based CF and Item-based CF, hybrid method makes much improvement based on values of both precision and RMSE.

We also list the precision for Content-based for the completeness of summary. It does not make sense comparing it with other recommenders. But it gives us an overall thought on all recommenders.

Table 1: Evaluation of all recommenders

	User-based CF	Item-based CF	Content-based	Matrix Factorization	Hybrid
Precision @ 5	0.3	0.16	0.63	0.07	0.4588
RMSE	1.0538	1.4146	N/A	1.1886	0.9221

VIII. Extra credit #2 (EC2): Hybrid Recommendation System

We combine content-based and item-based recommendation in our hybrid part. The idea is adding genre of the movies to the original user-item dataset to enrich item vectors. Then apply item-based recommendation on the new dataset. In Part C, we calculate items similarity based on users' rating. Using this hybrid method, we not only consider users' rating but also take the content of items (genre) into account, such that the similarity of items would be more accurate.

Step 1: Append the movie/genre table to the user-item-rating table.

	Movie1	Movie2	Movie3	...	Movie37
User1	4	4	4	...	NA
User2	5	5	5	...	NA
...					

User47	4	NA	NA	...	NA
Comedy	0	0	1	...	1
Action	0	1	0	...	0
...					
Drama	0	0	0	...	0

Step 2: Reshape the table to fit for the Mahout collaborative filtering recommender system as follows:

userID	itemID	Value
User1	Movie1	4
User2	Movie1	5
...	...	
User44	Movie37	5
Comedy	Movie1	0
Action	Movie1	0
...	...	...
Drama	Movie37	0

Step 3: Run item-based recommendation on this hybrid dataset, same as we did in part C.

Step 4: Evaluation. Since the matrix has both rating (1-5) and binary value, I would use both RMSE and precision value to evaluate this part. The output RMSE: 0.9221 and precision: 0.4588 which are better than those of user-based recommendation and item-based recommendation obtained in part EC1.