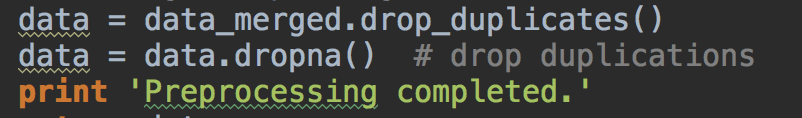
Modeling the Recommendation System

1. Data preparation

Before building a recommendation system, we need to clean and remodel the data set. Unnecessary columns in the data are removed to increase the efficiency of the training model process.

In the scoring data table, since we do not need to consider the user's comment time for the movie, so we get rid of the “timestamp” column.

Use the following commands to remove the duplicate and null values from the data set, respectively.



Finally, we could get a cleaner data set.

1. Modeling the Recommendation System

In this part we will use two different algorithms to build a recommendation engine for movies.

There are many approaches to build a recommendation engine. A popular approach for recommendation is collaborative filtering. User-based collaborative filtering is based on the assumption that users will probably like a product if it is liked by a similar user. A popular collaborative filtering algorithm is the k-Nearest neighbor algorithm, which calculates the similarity between the current user and all other users (based on rating behavior). While this algorithm is simple and intuitive, it is not useful for users and products with few data (sparsity). Also it can’t be used for new users and products (cold start). Other approaches include is content-based filtering and item-based collaborative filtering.

2.1 Item-based collaborative filtering

（1）Construction of Item similarity

In terms of item similarity, the implicit feedback data set has higher algorithm efficiency in calculating similarity, so we use implicit feedback dataset to construct the similarity matrix. Implicit feedback data refers to the data that can not clearly reflect the behavior of user preferences, the most representative of the hidden feedback data is the user's browsing behavior. The algorithm ignores the user's rating record for the movie and does not predict the user's rating for the unviewed video, but only focuses on the user's past browsing history of the movie. The film similarity calculation formula is shown in following formula.



The denominator |N(*i*)| denotes the number of users who liked the movie i, the numerator |N(*i*)∩N(*j*)| denotes the number of users who liked the movie i and the movie j at the same time.

There is a certain problem in above formula, if the film j is preferred by a lot of people, then the calculated similarity wij will be close to 1, indicating that many movies will have a high similarity with the popular movies. Based on the above analysis, we change the formula to the following one.



changed formula penalizes the weight of the movie j, thus reducing the likelihood that a popular movie will be similar to many other movies.



（2）Collaborative filtering recommendation

After building the movie similarity data set, users can be recommended based on the similarity degree and their historical behavior. First, we should select the movies which are rated higher by the certain user, thereby constructing a weighted list, including the other movies which are most similar to the selected movies. The following table is an example diagram based on item recommendations.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Movie** | **Rating** | **Sim.D** | **R.D** | **Sim.E** | **R.E** |
| **A** | 4.0 | 0.9 | 3.6 | 0.4 | 1.6 |
| **B** | 3.0 | 0.3 | 0.9 | 0.5 | 1.5 |
| **C** | 1.0 | 0.5 | 0.5 | 0.2 | 0.2 |
| **Sum** |  | 1.7 | 5.0 | 1.1 | 3.3 |
| **Normalization** |  |  | 2.94 |  | 3.0 |

As shown in the table, A, B, C are movies which are watched by the user. This user scored A, B and C for 4.0, 3.0 and 1.0. D and E are movies that have not been watched by the user. For videos that have not been watched, the corresponding column Sim.X will record how close it is to the movies being watched. For example, the similarity of D and A is 0.9, and Sim.D is the similarity between A and D, R.D is the evaluation value of the movie( 4.0 \* 0.9 = 3.6). The table predicts that this user may score a film D for 2.94 and a film E for 3.0. And so on, and ultimately in accordance with the forecast scores to recommend movies for this user.

2.2 User-based collaborative filtering

（1）Construction of User similarity

We use the same method to calculate the similarity between users, that is, the implicit feedback data set are used to calculate the user similarity.

|  |  |  |
| --- | --- | --- |
| Movie | User1 | User2 |
| A | 1 | 1 |
| B | 0 | 1 |
| C | 1 | 0 |
| D | 1 | 1 |
| E | 0 | 1 |
| F | 0 | 1 |

For example, User1 has watched A、C and D, User2 has watched A、B、D、E and F, so that the similarity between User1 and User2 equals to 2/() = 0.5164.

（2）Collaborative filtering recommendation

After building the user similarity data set, users can be recommended based on the similarity degree and their historical behavior. We will use the same algorithm which we used in IBCF to make recommendation for users.

1. Model Evaluation

We have construct the IBCF and UBCF recommendation models respectively. In order to select the more accurate prediction models, we need to test the two models. In the model evaluation, the data sets will be split into two parts, including training set and test set.

We write an algorithm to divide the original data set randomly into M copies(M = 8 in this experiment) according to the uniform distribution. One of them is the test set, and the remaining M-1 is the training set. The training set data is used to train the model, and then the testing index is tested by using the test data.

The recall rate, the accuracy rate and the coverage rate are used as the evaluation indexes of the model to test the two models respectively.

The recall rate describes the weighting of the user-rating record contained in the final recommendation list, as shown in following equation, where R(*u*) denotes the movie recommended for user u, and T(*u*) denotes the set of movies that the user likes on the test set.



The accuracy rate describes how many items in the recommended list are accurately predicted. The calculation formula is shown following. As with the variables in the recall, R(*u*) denotes the item recommended for user u, and R(*u*) denotes the set of items that the user likes on the test set.



Coverage reflects the ability of the proposed algorithm to mine long-tail items. The higher the coverage, the better the recommendation algorithm can recommend the items in the long tail to the user. The coverage indicates the proportion of items in the final recommendation list. If all items are recommended to at least one user, the coverage is 100%. The formula for calculating the coverage is shown as follows. Where |I| denotes the number of movies.

