Recommendation System: Principle and Application in Roommate Matching

Yang Yilin
ID: 1155112516
1155112516@link.cuhk.edu.hk

Chen Yangchun
ID:1155121611
1155121611@link.cuhk.edu.hk

ABSTRACT
This paper describes the current recommended methods, which are usually divided into the following three main categories: collaborative filtering, content-based and hybrid recommendation approach. Then, we would focus on neighborhood-based collaborative filtering recommendations, highlighting its user-based and item-based classification, and introducing its steps and four algorithms for calculating similarity. This paper also practices the theory, using Python, with the user-based collaborative filtering recommendation algorithm, here is the most widely used cosine similarity algorithm to help the S. H. Ho College of Chinese University of Hong Kong to match the four-person dormitory for students who have been selected as eligible accommodation.

Keywords
Recommendation system, collaborative filtering, distributed database system, roommate matching

1. INTRODUCTION
The purpose of a recommendation system is to provide meaningful recommendations to a collection of users for items or products that might interest them. The recommendation system can meet the dual needs of users and firms, eliminating information barriers and increasing information value, enhancing user experience and stickiness, and providing services and products with design directions.

It is difficult for users to describe their needs and ideas with appropriate keywords, or to describe information that they are not aware of but may be of interest. At the same time, users may face a large amount of irrelevant data when searching, which may confuse their choices.[1]Therefore, the recommendation system searches the database and presents it to the user so that they can find what they want in a shorter time and with significantly higher precision.

Also, firms can collect large amounts of data for a deeper analysis of users' behavior, then automatically generates recommendations to users.[2]

Since the recommendation system can actively collect the user's characteristic data and analyze the user's preferences, the user can customize and provide information of interest to the user. At the same time, the recommendation system can track the changes in users' needs in time, and automatically adjust the way and content of information services according to changes. Different from the "one-to-many" information service provided by search engines, the results of the recommendation system are more in line with the individual needs of users, realizing "one-to-one" information services, and the degree of user participation is also lower. Greatly reduce the cost of users searching for information

In the remainder of this paper, we describe the principle of recommendation system and distributed DBMS related content. We then present the application of recommendation system in roommate matching, and conclude with a discussion.

2. RELATED WORK
2.1 Comprehensive Survey
Adomavicius G and Tuzhilin A (2005) describe recommended methods that are generally classified according to content-based, CF, Hybrid and their limitations, and discuss possible extensions that can improve recommendation and make the recommendation system suitable for a wider range of applications. Then it also introduces the characteristic indicators that need to be paid attention to when designing the engine. [3]

JONATHAN L. HERLOCKER et al. (2004) proposed two questions about the recommendation algorithm: whether it is worth taking the time to study the recommendation algorithm; whether all the algorithms are equally good. Then they came up with an evaluation experiment: not all metrics yield the same recommendation, and the correct grouping of metrics is likely to affect the accuracy of the recommendation. [4]

Stuart E. Middleton and David C show how a physically existing external entity can handle the cold-start problem in the recommendation system: a very detailed business assessment of the validity of the individual data in the recommendation system is required. They suggested that the recommendation system could not provide too many valuable recommendations before new users filled in their profile.[5]

2.2 Collaborative Filtering Survey
Koren Y and Bell R introduced some important breakthroughs in collaborative filtering in recent years, including factorization, time-related recommendations, recommendations based on neighbors, and integration of multiple methods.[6]

Su X, Khoshgoftaar TM introduced various collaborative filtering methods and evaluation criteria according to the classification method of memory-based, model-based, hybrid, and compared the effects of evaluation based on Netflix data.[7]

Sonali R. Gandhi and Jaydeep Gheewala gave an overview of CF and its problems and then talked about the implementation of MapReduce. They cited a paper to address the issues of scalability and sparsity and the use of machine learning methods. They then built a hybrid approach that combines dimensionality reduction techniques with clustering methods to improve traditional user-based CF performance.[8]

Thomas Hofmann introduced a model-based collaborative filtering algorithm in a latent semantic model. Behind this model subconsciously assumes that the user's preferences are distributed as a vector with some potential factors. In addition, their experiments show that their algorithms are very accurate and have low time complexity.[9]

2.3 Content-based and Hybrid Approach Survey

Pazzani MJ and Billsus D provide a macro view of content-based strategic architecture and application areas, including referral pages, news articles, restaurants, TV shows, and items for sale.[10]

Burke R introduces a framework for the fusion of multiple recommended algorithms. They surveyed the actual and possible mixed recommenders and introduced a novel hybrid, EntreeC, a system that combines knowledge-based recommendations and collaborative filtering to recommend restaurants.[11]

2.4 Distributed Database System Survey

Hosein Jafarkarimi1, Alex Tze Hiang Sim and Robbab Saadatdoost proposed a recommended model for mining transaction databases to reduce response time. They then applied and evaluated their models at the Malaysian Polytechnic University and the results reflect a high degree of accuracy.[1]

Sandeep Kulkarni proposes a hybrid logic clock HLC that combines the best logic clocks with physical clocks. They then showed that HLC has many advantages for sorting without wait transactions and performing snapshot reads in a multi-version global distributed database.[12]

3. RECOMMENDED SYSTEM TYPES

3.1 Collaborative Filtering Recommendations

Collaborative Filtering (CF) systems work by collecting user feedback in the form of ratings for specified projects, and when determining how to recommend projects, work with the similarity of rating behaviors between several users.[2] Users will be recommended items that people with similar tastes and preferences liked in the past. This is quite similar to the word-of-mouth in real life.

In terms of scenarios, Facebook, WeChat and other social networking companies use collaborative filtering to recommend friends, groups, and more. Twitter and Weibo recommend followers to users. Amazon, Netflix, Taobao, and Jingdong recommend items that users might be interested in.

The collaborative filtering recommendation algorithm is divided into two categories, namely neighborhood-based recommendations and model-based recommendations respectively. And neighborhood-based recommendation techniques is also called as a memory-based approach.[2]

3.1.1 Neighborhood-based Techniques

The recommendation mechanism for neighborhood-based collaborative filtering is the most widely used today. It generally finds a small part of a large number of users similar to the user's taste. In collaborative filtering, these users become neighbors and then generate recommendations based on their preferences.

3.1.1.1 Advantages

It is easy to implement and easy to add new data. Collaborative filtering of user preference information is easy to collect, using a distributed database. Moreover, the procedure of the collaborative filtering recommendation algorithm is not complicated, so the implementation of the entire collaborative filtering system is relatively easy.

The recommended object can be any type of resource. Since collaborative filtering generates recommendations based on similar users' interest preferences, its key is to find neighbor users that are similar to the target user preferences without having to analyze and extract content information. At the same time, it does not require strict modeling of the item or user and does not require that the description of the item be machine understandable.[13] This allows collaborative filtering techniques to recommend any type of resource item to the user.

It has less interference to users. Collaborative filtering recommendation generally uses the scoring vector as a representation of user preferences. After logging in to the system, users only need to score some items to obtain recommended services. The user is less affected during the whole process of using the system.

The recommendations it calculates are open. Collaborative filtering does not compare the correlation between project content and user description. The recommended project is not completely limited to the user's historical preferences and can share the experience of others. This will help to identify potential interests of users, implement jump recommendations, and support users to discover potential interest preferences.

3.1.1.2 Disadvantages

It is limited scalability for large datasets. With the increase in the number of users and projects, the computational complexity of the algorithm increases sharply. This may seriously affect the real-time performance of the system recommendation. This leads to the scalability problem of collaborative filtering.

Performance decrease when data are sparse.[13] The actual number of users and projects in the website is huge, and users usually only score a small number of items. The recommended effect depends on how much and accuracy the data is. The data that can be used to calculate the similarity between users/projects is very limited, making the nearest neighbor search less accurate and the recommendation quality is poor. It is often seen that there is no common rating between two users, which makes that similarity cannot be calculated. Even if the user similarity can be calculated, the reliability is difficult to guarantee. In most implementations, user history preferences are stored using sparse matrices, while computations on sparse matrices have significant problems, including the potential for a small number of people to have a large impact on the accuracy of recommendations.

Also, for some special taste, users can not give a good recommendation.

3.1.2 Model-based Techniques

Model-based recommendations are a typical machine learning problem. The method is to train a recommendation model based on the user preference information of the sample. It can then predict and calculate recommendations based on real-time user preferences.

Typical model-based collaborative filtering includes collaborative filtering based on clustering techniques, collaborative filtering based on probabilistic methods, and collaborative filtering based on matrix decomposition.

3.2 Content-based Recommendation

The core idea of content-based recommendations is to recommend similar items to users based on their previous preferences. [2]It has a premise that the same user has a common interest or attraction for the same type of item and the preference does not change over time.

The content-based recommendation algorithm has only one key point - the tag. The recommendation algorithm breaks down the product into a series of tags and describes the user as a series of tags based on the user's behavior (eg, purchase, browsing).

Figure 1 shows the basic principles of content-based recommendations. First, we need to model the metadata and label each item. The similarity between the items is then compared by the metadata set, and the items 1 and 3 whose labels are "Tag 1" are considered to be similar items. Finally, the recommendation is implemented. For user A, he likes item 1, and the system can recommend item 3 to him.

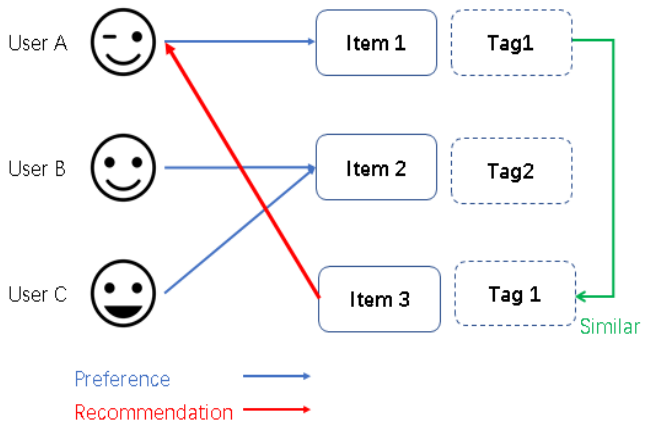

Figure 1 Content-based recommendation.

3.2.1 Advantages

It can model the taste of the user and provide more accurate recommendations. Each user's recommendation is based on the user's own behavior, and the user is independent.

3.2.2 Disadvantages

But it also has the following problems. It requires analysis and modeling of the item, and the quality of the recommendation depends on the completeness and completeness of the item model. The analysis of the similarity of an item depends only on the characteristics of the item itself and does not take into account the attitude of the person to the item. Because it is necessary to make recommendations based on the users' past preference history, there is a cold-start problem for new users.

3.3 Hybrid Approaches

These methods combine both collaborative and content-based approaches. [3]

3.3.1 Weighted Hybridization

The two recommendations are combined according to a certain weight by a linear formula. The value of the specific weight needs to be repeatedly tested on the test data set to achieve the best recommendation.

3.3.2 Switching Hybridization

This allows the selection of the most appropriate recommendation mechanism to calculate recommendations in different situations.

3.3.3 Mixed Hybridization

Both recommendation mechanisms are adopted, and their recommendation results are displayed to the users in different areas. In this way, users can get very comprehensive recommendations and easier to find what they want.

4. Neighborhood-based collaborative filtering

Collaborative filtering is a typical method of harnessing collective intelligence. A simple case can help to understand Collaborative Filtering, short for CF. If you want to watch a movie, but you don't know which one to choose, what do you do? Most people will ask friends around to see if there are any worth-watching movie recently. And we generally prefer to get recommendations from friends with similar tastes. This is the core idea of collaborative filtering.

4.1 User-based Collaborative Filtering

The basic principle of user-based collaborative filtering recommendation is to find the "neighbor" user group with similar taste to that of the current user and based on the historical preferences of all users for different items.

The data of user's historical behavior such as product purchase, collection, content review or sharing are used to find the different level of user's preference of the product, which the similarity between users is calculated based on. Product recommendations are made between users with the similar preferences.

The basic principle of the user-based collaborative filtering recommendation shows below in the figure. User A likes item 1, item 3; user B likes item 2; user C likes item 1, item 3 and item 4. From the historical preference information of these users, we can find user A and user C share the similar taste as they have 2 same preference for Item 1 and Item 3. So, item 4, which User D likes but User A not yet likes, can be recommended to the user A.

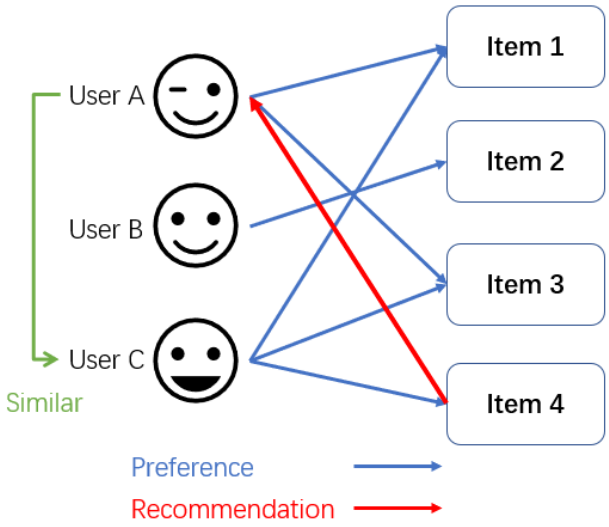

Figure 2 User-based collaborative filtering

Based on “neighbor” user groups, there are two mechanism, which are User-based collaborative filtering recommendation and the traditional demographic-based recommendation. But they differ in the way how to calculate user similarity. The traditional demographics-based mechanism only considers the characteristics of the user itself, such as the age, the occupation, the interest of the user, while the user-based collaborative filtering mechanism calculates the user's similarity on the user's historical preference data. The basic assumption is that users who like similar items may have the same or similar tastes and preference.

4.2 Item-based Collaborative Filtering

Item-based collaborative filtering recommendation is based on the similarity of items. Because the same user or multiple user within a neighborhood have the same preference for items they choose. It can be inferred that the items the same user or multiple user within a neighborhood prefer share some similarity. Then the similar items are recommended to the user based on the user's historical preference information.

Since the item-based collaborative filtering recommendation still calculates the similarity of the item based on the historical preference of the neighbor users, it is also a kind of neighbor collaborative filtering. It is different from the content-based recommendation, which is based only on the attribute characteristics of the content itself.

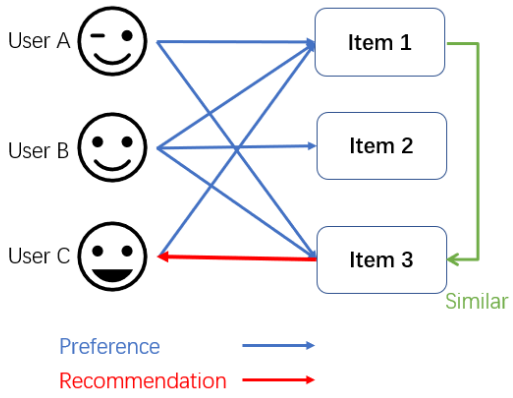

Figure 3 Item-based collaborative filtering

4.3 Comparison

The User-based algorithm has two problems. On the one hand, data sparsity, also known as cold-start problem. A large e-commerce recommendation system generally has lots of items, but less than 1% of them that the user may buy. So the overlap between the items purchased by different users is low, resulting in the algorithm unable to find a user's neighbor, that is, the similar users. On the other hand, the algorithmic scalability. The calculation of the nearest neighbor algorithm increases as the number of users and items increases. It is not suitable for use in situations where the amount of data is large.The basic idea of Iterm-based is to calculate the similarity between items based on the historical preference data of all users in advance, and then recommend the similar items to the user. Due to the relatively fixed similarity of the items compared with users, the similarity between different items can be calculated offline in advance, and then search the results stored when the recommendation is made, partial solving both problems of User-based collaborative filtering.

4.4 Collaborative Filtering Implementation Steps

4.4.1 Data Collection

4.4.1.1 Data Acquisition

Uses customized data acquisition programs, or use the open source framework FLUME to collect user behaviors such as likes, comments, ratings, shares, etc.

FLUME is a distributed log collection system that collects data from various servers, making simple processing and sends it to the designated place. Flume's Agent is mainly composed of three components: source, channel and sink. Receive data from the data generator and pass the received to one or more channels. Channel as a short-lived storage container that will receive the data from the source until they are consumed by the sinks. It plays a role as a bridge between source and sink. Channel is a complete transaction, which guarantees the consistency of the data when it is sent and received. And it can be linked with other sources and sinks. The sink stores the data in a centralized memory such as HDFS(Hadoop Distributed File System)

4.4.1.2 Data Preprocessing

Cast the data processing can into Map and Reduce operations, get the regular data into HDFS.

When dealing with a large number of data, MapReduce can partition the data input, schedule the execution among different machines, and then write the data in an efficient way. It can Handling machine failures and manage required inter-machine communication.

The step of map is to get the selected data while discard the useless and group the data by key Sort and Shuffle. The step of Reduce is to aggregate or transform the data to get the result then write it.

4.4.1.3 Data Warehouse Technology and ETL

Based on Hive on Hadoop, the regular data can be mapped into a table, and then be extracted, transformed and loaded (ETL, Extract-Transform-Load) to a form which can be used to calculate the similarity in the later step.

Hive is a data warehousing tool based on Hadoop that maps structured data files into a database table and provides a simple sql query function. Hive defines a simple SQL query language called HQL, which is easier for user familiar with SQL to use. At the same time, the language also allows a user-defined Mapper and Reducer handle the complex analysis work. And ETL is a mechanism for storing, querying, and analyzing large-scale data stored in Hadoop.

4.4.2 Similarity Calculation

There are 4 common ways to calculate the similarity.

4.4.2.1 Euclidean Distance

In the user-based collaborative filtering recommendation, the user's preference for different commodities is set as a vector. The number of items is the dimension of the vector. By calculating the multi-dimensional distance between the two users' preferences for different commodities, the similarity of users can be get. The closer the distance, the higher the similarity. When there are only two products, it is the distance between the two points on the two-dimensional plane. The principle is the same in the multi-dimensional expansion. The item-based collaborative filtering recommendation can also use this function by setting the names of the different users who like the item as vector. The more

consistent of the users who like the item, the closer the Euclidean record is, the more similar the item is.

$$d(x, y) = \sqrt{\sum (x_i - y_i)^2} \quad (1)$$

$$\text{similarity } (x, y) = \frac{1}{1 + d(x, y)} \quad (2)$$

4.4.2.2 Pearson Correlation Coefficient

The Pearson correlation coefficient is used to measure the linear correlation between two variables. It is a linear regression for two users' preferred items, or users who prefer the same item. The similarity can be judged the fitting effect. It can be applied when there is data expansion with one large variable and the other small one, while the two variables still have a significant linear relationship. The Pearson coefficient is the slope in the linear regression analysis. The Pearson coefficient is [-1, 1]. when it is equal to -1, the similarity shows a complete negative correlation. When it is equal to 1, the similarity degree is completely positively correlated. When it is 0, two Users or items are not relevant.

$$r = \frac{N \sum x_i y_i - \sum x_i \sum y_i}{\sqrt{N \sum x_i^2 - (\sum x_i)^2} \sqrt{N \sum y_i^2 - (\sum y_i)^2}} \quad (3)$$

4.4.2.3 Cosine Similarity

This method sets the user's rating for different goods as a vector. The number of items is the dimension of the vector. The similarity between the items is calculated by calculating the angle cosine between the two vectors. The smaller the angle between the two vectors, The higher the similarity. The item-based collaborative filtering method can also use this method by setting the users name as the vector.

$$T(x, y) = \frac{x \cdot y}{\|x\|^2 \times \|y\|^2} = \frac{\sum x_i \sum y_i}{\sqrt{\sum x_i^2} \sqrt{\sum y_i^2}} \quad (4)$$

4.4.2.4 Tanimoto Coefficient

The Tanimoto coefficient is the union of two sets divided by the intersection of the two sets. For example, it uses the same items two people preferred divided by all preferences they have. This shows the degree of correlation between two people. Because generally the more the same preferences they have, the similarity they share.

$$T(x, y) = \frac{x \cdot y}{\|x\|^2 + \|y\|^2 - x \cdot y} = \frac{\sum x_i \sum y_i}{\sqrt{\sum x_i^2} + \sqrt{\sum y_i^2} - \sum x_i \sum y_i} \quad (5)$$

5. Recommendation System Application

In the following, it is the roommate matching of S. H. Ho College of Chinese University of Hong Kong by using recommendation system.

As in the actual, candidates applying the accommodation will be matched into different room according to their similarity of interests

In this case, we simulate the list of participating students for different types of activities to sum up 10 different kinds of interests and the level of the interest of the candidates who meet the accommodation conditions. Then we match the 40 roommates for each 4-person room by calculating the cosine similarity of their interests.

5.1 Data Collection

We simulated the data of the distributed database system to get the data source of this roommate matching recommendation system.

Suppose the school has different activities, swimming competitions, chess competitions, dance competitions, dance viewing sessions, film production competitions and movie viewing activities. The activity records are stored in different data sheets. We have summarized 10 kinds of interest in the activities, namely, swimming, chess, Piano, dancing, violin, movies, music, basketball, football and badminton. These 10 kinds of interest are the basis of different preferences for roommate matching.

Table 1 Data collection

No.	Interest
1	swim
2	chess
3	Piano
4	dance
5	violin
6	movie
7	music
8	basketball
9	football
10	badminton

The same interest may contain different activities, such as a dance interest. it may comes from having a dance competition, or attending a dance-like viewing. From the activity records of different activities, we can learn the frequency of different students' participation in different activities and the awards. It can be used to draw different students' preference scores for different interests. In the data file, the data is simulated into the following state, including the first One student ID, the second interest ID, the third rating of interest, and the fourth is the timestamp of the data.

5.2 Data Processing

5.2.1 Raw Data

Table 2 Raw data

User_id	Hobby_id	rating	timestamp
1	1	5	1541905871
2	7	4	1544497271
3	1	4	1543978871

This data needs to be converted into a data structure, which is formatted into a list.

List[[user id, interest id, interest rating]...]

This list can be represented as:

Table 3 Data list

	Hobby_id_1	Hobby_id_2	Hobby_id_3	…	Hobby_id_n
User_id_1	5	3	4		
User_id_2	4				
User_id_3	4	2	3		
…					
User_id_m					

The case assumes 10 interests, and there are 10 Hobby_ids on the abscissa.

If there are 100w students, the ordinate has 100w user_id. This case assumes 40 students.

The content score in the table represents the user's evaluation of interest. The more you like the higher the rating. If there is no

rating of the interest from a student, it means that the interest is not evaluated.

5.2.2 Vectorization and Dictionaryization

After the original data format is a list, a vector of M rows and N columns is obtained, and each user is a 1*N vector. In collaborative filtering, a very important part is to calculate the similarity of users, which can be abstracted to calculate the similarity between vectors. The specific method can have cosine similarity theorem, Euclidean theorem and so on.

The purpose of dictionary is to facilitate the searching. There are three places in this dictionary:

1). User dictionary: test_dic[user id]=[(interest id, interest rating)...]

It is used to find interest by student ID.

2). Interest dictionary: test_item_to_user[interest id]=[user id1, user id2...]

It is used to find users by interest ID.

3). Neighbor dictionary: user_rate_dic[user id]=[(distance, user, neighbor id)...]

It is used to find neighbors by user ID.

Table 4 Neighbor dictionary

	Hobby_id_1	Hobby_id_2	Hobby_id_3	…	Hobby_id_n
User_id_1	5	3	4		
User_id_2	4				
User_id_3	4	2	3		
…					
User_id_m					

For example, in the Table 4, we can use the user dictionary to find the student's interest in User_id_1.

5.3 Student Similarity Calculation

To calculate students' similarity, we need to find the user's neighbor first, such as User_id_2 and User_id_3 and User_id_1 in the table.

It is possible to for students to become neighbors only if they share at least a common interest.

Table 5 Student similarity calculation

	Hobby_id_1	Hobby_id_2	Hobby_id_3	…	Hobby_id_n
User_id_1	5	3	4		
User_id_2	4				
User_id_3	4	2	3		
…					
User_id_m					

The way we used to calculate the similarity is by calculating the Cosine correlation coefficient.

5.4 Roommate Matching

The general recommendation system only recommends the item to the user, so the item can be repeatedly recommended. Unlike the general recommendation system, in the roommate matching, each roommate can only be arranged in one room. And the actual situation is, for student 1, his closest neighbor is student 2. But for student 2, his closest neighbor is not student 1, but other students. So, we introduce similarity ranking and greedy algorithms to recommend the most similar roommates.

5.4.1 Similarity Ranking

We rank the results of the similarity calculations between all two students and rank them in descending order. In the table, if User_id_3 and User_id_1 are more similar. The code for sorting is as follows. We can think that User_id_3 and User_id_1 have the same interests, so first arrange Student 1 and Student 3 in Dormitory 1. And later arranged the dormitory according to the greedy algorithm.

5.4.2 The Greedy Algorithm

The greedy algorithm is to solve the problem that the similarity between students is not mutual. For example, for student 1, User_id_1 and User_id_3 are the closest neighbors. But for student 3, User_id_3 and User_id_1 are the closest. It is necessary to develop a certain strategy for User_id_3 who should be recommended to become a roommate. The greedy algorithm means that in each step of solving, it requires "greedy" to choose the best operation, and hopes to generate an optimal solution of a problem through a series of optimal choices. The greedy algorithm must meet the following conditions at each step:

1). Feasible: that is, it must satisfy the constraints of the problem.

2). Local optimum: He is the best local choice of all feasible options in the current step.

3). Arrangement cannot be canceled: that is, once the choice is made, the steps in the algorithm cannot be changed

The usage of the greedy algorithm in this case is:

1). Rank the similarity of two users

2). Take a largest similarly group and put it in the first dormitory. Mark both people as selected or removed from the list

3). Then try to find any larger similarity groups with the arranged students contain. If the other of the larger similarity groups is not arranged in any dormitory, then put him in the dormitory and mark "arranged" to delete him in the ranking table. If the neighbor has been arranged in other dormitory, then find the next larger similarity group.

4). Repeat 3 for work until a dormitory is full.

5). Continue with step 2 and select the most similar group of remaining data.

So, using the greedy algorithm, if the similarity of the four students is as shown in the table, then the final recommendation is

User_id_1 and User_id_2 one dormitory

User_id_3 and User_id_4 one dormitory

Table 6 Final recommendation

User_id_1	User_id_2	0.9
User_id_1	User_id_3	0.8
User_id_1	User_id_4	0.7
User_id_2	User_id_3	0.6
User_id_2	User_id_4	0.5
User_id_3	User_id_4	0.4

Programming code of the greedy algorithm shows below.

```
a=len(all_neighbors_dis)
for m in range(a):
    nearest_neighbors_dis=all_neighbors_dis.pop(0)
    roommates=[]
    roommates_num=4
    #取用户和邻居添加到数组中
    neighbors_id1=nearest_neighbors_dis[1]
    neighbors_id2=nearest_neighbors_dis[2]
    if neighbors_id1 in killneighbors or neighbors_id2 in killneighbors:
        continue
    killneighbors.append(neighbors_id1)
    killneighbors.append(neighbors_id2)

    roommates.append(neighbors_id1)
    roommates.append(neighbors_id2)
    neighbor_id_next=neighbors_id2
    for i in range(roommates_num-2):
        for item in user_rate_dic[neighbor_id_next]:
            if item[2] not in killneighbors:
                killneighbors.append(item[2])
                neighbor_id_next=item[2]
                #且加到宿舍，第三个/第四个
                roommates.append(neighbor_id_next)
                break
```

(cont.)

```
    table = Texttable()
    table.set_deco(Texttable.HEADER)
    table.set_cols_dtype(['t','t','t','t']) # automatic
    table.set_cols_align(["l", "l", "l", "l"])

    rows.append([roommates[0],roommates[1],roommates[2],roommates[3]])
table.add_rows(rows)
    #取下一个贪心最大值
```

Figure 4 Programming code of the greedy algorithm

5.5 Roommate Matching Result

Take the first 4-person roommate matching as example.

After the ranking of similarity, student 40 and student 32 have the largest similarity than all other students. The print result of the neighbor dictionary of student 40 shows below.

```
user_rate_dic[40]: [(1.0, 40, 32), (0.9715642478872429, 40, 33), (0.967091360980
4876, 40, 34), (0.9655180288004046, 40, 6), (0.9637388493048533, 40, 20), (0.954
8265766308701, 40, 36), (0.9537260442854445, 40, 14), (0.9524301680017466, 40, 1
6), (0.9510974737567147, 40, 31), (0.9498342019696734, 40, 24), (0.9496532203248
237, 40, 25), (0.9471690380301031, 40, 9), (0.9450254584387164, 40, 29), (0.9384
89160756017, 40, 10), (0.9383148632568366, 40, 3), (0.9342914068428719, 40, 19)
, (0.9310272428781118, 40, 12), (0.9309493362512627, 40, 38), (0.927807946833231
4, 40, 18), (0.9270248108869579, 40, 13), (0.9222219343686804, 40, 8), (0.916911
8129500092, 40, 30), (0.914769847607588, 40, 23), (0.9104654680003259, 40, 35),
(0.9065910942051448, 40, 11), (0.9058216273156765, 40, 2), (0.9035163690190994,
 40, 4), (0.9, 40, 27), (0.8966226649989563, 40, 7), (0.8859855473057758, 40, 28)
, (0.8838834764831844, 40, 26), (0.872509475832402, 40, 37), (0.871203479245162
2, 40, 39), (0.865410983413526, 40, 15), (0.8612563219548984, 40, 5), (0.8482760
119874715, 40, 1), (0.8458410929196717, 40, 17), (0.8426509759922837, 40, 22), (0
.7877263614433762, 40, 21)]
```

Figure 5

Then, we find the neighbor dictionary of student 32. The relatively larges similarity student 32 have with is student 39. The print result of the neighbor dictionary of student 39 shows below. So we arrange student 39 in this dormitory with there are student 40 and 32 already.

```
user_rate_dic[32]: [(1.0, 32, 40), (1.0, 32, 39), (1.0, 32, 38), (1.0, 32, 37),
(1.0, 32, 35), (1.0, 32, 30), (1.0, 32, 29), (1.0, 32, 28), (1.0, 32, 27), (1.0,
 32, 25), (1.0, 32, 24), (1.0, 32, 23), (1.0, 32, 22), (1.0, 32, 21), (1.0, 32,
20), (1.0, 32, 19), (1.0, 32, 17), (1.0, 32, 16), (1.0, 32, 15), (1.0, 32, 14),
(1.0, 32, 13), (1.0, 32, 12), (1.0, 32, 11), (1.0, 32, 10), (1.0, 32, 9), (1.0,
32, 8), (1.0, 32, 7), (1.0, 32, 6), (1.0, 32, 4), (1.0, 32, 3), (1.0, 32, 2), (1
.0, 32, 1)]
```

Figure 6

Check the neighbor dictionary of student 39, to find out student 11. So there are student 40, student 32, student 39, and student 11 in this first room.

```
user_rate_dic[39]: [(1.0, 39, 32), (0.9787120517625908, 39, 11), (0.977590055598
7752, 39, 26), (0.9737289911202953, 39, 22), (0.9729936782611747, 39, 9), (0.970
8119487587379, 39, 12), (0.9707877788745956, 39, 33), (0.968424925767582, 39, 14
), (0.9643063789340139, 39, 30), (0.963356865536248, 39, 23), (0.961280786970633
7, 39, 10), (0.9594020884180302, 39, 24), (0.9581078238805528, 39, 25), (0.95022
25941594079, 39, 18), (0.9499507445716855, 39, 35), (0.9476976527994018, 39, 19)
, (0.9444444444444444, 39, 28), (0.9438798074485389, 39, 6), (0.942972009950318
9, 39, 29), (0.9421096069851023, 39, 38), (0.9420265190721552, 39, 13), (0.941706
6764113289, 39, 21), (0.9390678560679474, 39, 27), (0.9381226327480188, 39, 4),
(0.9338340252957616, 39, 8), (0.9333809511662426, 39, 7), (0.9298547766256364, 3
9, 36), (0.9271303370595978, 39, 5), (0.9262313996235813, 39, 16), (0.914702166 5
807529, 39, 17), (0.9079593845004517, 39, 20), (0.9077415526644965, 39, 3), (0.9
063269671749657, 39, 37), (0.9041944301794651, 39, 31), (0.8712034792451622, 39,
 40), (0.8665782448262421, 39, 2), (0.8600662598876552, 39, 1), (0.807130922258 3
08, 39, 15), (0.7624928516630234, 39, 34)]
```

Figure 7

Same as above, we the roommate matching for all 40 students shows below.

neighbor1	neighbor2	neighbor3	neighbor4
40	32	39	11
5	2	35	24
29	18	20	33
8	4	7	38
25	19	12	23
27	3	36	34
14	6	16	17
26	22	9	10
30	28	21	1
37	31	13	15

Figure 8

6. REFERENCES

[1] Hosein Jafarkarimi1, Alex Tze Hiang Sim, Robab Saadatdoost. 2012. A Naïve Recommendation Model for Large Databases[J]. International Journal of Information and Education Technology, Vol. 2(3).

[2] Peem Melville, Vikas Sindhwani. 2010. Recommender Systems[J]. Encyclopedia of Machine Learning Chapter No: 00338 Page Proof 2010(1):1.

[3] Adomavicius G, Tuzhilin A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions[J]. Knowledge and Data Engineering, IEEE Transactions on, 17(6): 734-749.

[4] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. 2004. Evaluating Collaborative Filtering Recommender Systems[J]. ACM Transactions on Information Systems, Vol. 22(1): 5–53.

[5] Stuart E. Middleton, David C. De Roure and Nigel R. Shadbolt. Capturing knowledge of user preferences: ontologies in recommender systems[J]. K-CAP '01 Proceedings of the 1st international conference on Knowledge capture: 100-107.

[6] Koren Y, Bell R. 2011. Advances in collaborative filtering[M]. Recommender Systems Handbook, Springer US, 145-186.

[7] Su X, Khoshgoftaar T M. 2009. A survey of collaborative filtering techniques[J]. Advances in artificial intelligence, 2009: 4.

[8] Sonali R. Gandhi, Jaydeep Gheewala. 2017. A survey on recommendation system with collaborative filtering using big data[J]. International Conference on Innovative Mechanisms for Industry Applications (ICIMIA).

[9] Thomas Hofmann. 2003. Collaborative filtering via gaussian probabilistic latent semantic analysis[J]. I Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, 259−266.

[10] Pazzani M J, Billsus D. 2007. Content-based recommendation systems[M]. The adaptive web. Springer Berlin Heidelberg, 325-341.

[11] Burke R. 2002. Hybrid recommender systems: Survey and experiments[J]. User modeling and user-adapted interaction, 12(4): 331-370.

[12] Sandeep Kulkarni. 2014. Logical Physical Clocks and Consistent Snapshots in Globally Distributed Databases Principles of Distributed Systems[J]. 18th International Conference, OPODIS 2014, Cortina d'Ampezzo, Italy, December 16-19, 17-32.

[13] Xiaoyuan Su, Taghi M. Khoshgoftaar. 2009. A Survey of Collaborative Filtering Techniques[J]. Advances in Artificial Intelligence, 19.