COMP9417 Topic 3.4: Recommender system using collaborative filtering

Team Name: Pokemon
Team Members: z5122763 Yirong CHEN
z5085901 Jinjie JIN
z5119536 Jinlong ZHANG

Introduction

The amount of data is growing exponentially nowadays, it is estimated that at least 2.5 quintillion bytes of data is produced every day. In terms of the internet-based media platforms and services , The ‘Big Data’ company, Youtube,for example, experiences the about 4.14 million videos watched per minute. Another case is that the Netflix has seen a 20 percent decrease in the amount of the movies compared to 2017. These is the situation where a user interacts with a large catalog of items (i.e the products at Amazon or movies at Netflix), the matter is that when the user have a large catalog of the items, the user does not know exactly what they looking for and this is why the recommender system come in. The recommender system is a system recommends certain items that the user wound be interested in to the user.There are two basic kinds of algorithms that the recommender system use. The first algorithm is called content-based filtering which relies upon the similarities between the items themselves (i.e. between two movies or two songs). The other algorithm is called collaborative filtering which is used in this project.This algorithm relies upon not the qualities of the object itself but how the users responded to these same objects.There are two techniques called user-based and item-based collaborative filtering. The user-based collaborative filtering recommends items to the user based on the combination of the user’s behaviors and other users’ behaviors.The another technique is called the item-based collaborative filtering. Instead of the comparison of the different users,The basic idea idea of this algorithm is the comparison of different items.

The goal of this project is to campare two recommender system using two different types of collaborative filtering: user-based and item-based base on the RMSE. The features used to compared are the different user similarity measure methods and the varied number of the similar users. With the comparison, the most suitable similarity measure method and number of the similar user is chosen and apply in the final recommender system to minimize the RMSE.

Implementation

Methods used in this particular project are user-based collaborative filtering and item-based collaborative filtering.

User-based Collaborative Filtering
The basic idea of user-based collaborative filtering is that given a user, the algorithm will try to find other users that are similar to that user. The detailed progress of the recommender system using collaborative filtering is shown below:

1. *The creation of the matrix storing the users' ratings for the movies*
   The input data of the user-based collaborative filtering is a m*n matrix. The table 1 below is an example with 401 users and a bunch of movies. The value of the matrix[i][j] is the rating given by user i for movie j, for example, the User 1 has given the ratings 4 ,4 and 5 to the movie 1, movie 5 and movie 20 respectively.

	Movie1	…	Movie5	...	Movie 20	…
User 1	4	…	4	...	5	...
User 2	3	…	4	...	3	...
User 3	4	...	3	...	5	...
…	...					
User 400	3	...	5	...	3	...
User 401	2	...	2	...	4	…

Table 1 ratings matrix

2. *The calculation of the user similarity matrix*
   In order to calculate the similarity matrix, it is straightforward to compared the given user to all the other users.However, this is time consuming because for example,there would be no same movie viewed between User1 and User2 in reality .In this case, there is no need to conduct similarity calculation between User1 and User2 and the similarity between User1 and User2 is 0.
   Therefore, the movie_user matrix is create to solve this issue.
   The table 2 below is an example for the movie_user matrix, each row records a list of users who have seen the same model.

Movie 1	User 1, User 2 … User 390
Movie 2	User 3, User 4 … User 200
Movie 3	User 1, User 7 … User 330
…	...
Movie 5600	User 1, User 9 … User 310

Table2 movie_user matrix

The user_related(table 3) matrix is then generated with the movie_user matrix.The user_related is a m*m matrix with values 1 and 0, the value 1 in matrix[i][j] means there is at least one same movie viewed between User i and User j and the value 0 in matrix[i][j] means there is not same movie viewed between User i and User j. It is noted that the value of matrix[i][i] is 0.

	User 1	Use 2	...	User 400	User 401
User 1	0	1		0	1
User 2	1	0		1	0
...	...				
User 400	1	1	...	0	1
User 401	1	0	...	1	0

Table 3 user_ralated matrix

The last step is to calculate the similarity matrix and there are three formulas which can be used to calculate the the user similarity matrix.The formulas are shown below:

Jaccard Similarity Formula:

$$sim(i,j) = \frac{N(i) \cap N(j)}{N(i) \cup N(j)}$$

$$(where\ the\ N(i)\ and\ the\ N(j)\ is\ the\ number\ of\ the\ number\ of\ movies\ user\ i\ and\ user\ j\ have\ views\ respectively)$$

Cosine similarity:

$$sim(i,j) = cos(i,j) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 \cdot \|\vec{j}\|_2}$$

$$(where\ the\ i,j\ is\ the\ vector\ for\ user\ i\ and\ user\ j\ respectively\ )$$

Euclidean distance:

$$dis(i,j) = \sqrt{\sum (i_m - j_m)^2}$$

$$(where\ the\ i,j\ is\ the\ vector\ for\ user\ i\ and\ user\ j\ respectively\ )$$

$$sim(i,j) = \frac{1}{1 + dis(i,j)}$$

Below(table 4) is a example of the user similarity matrix using Cosine similarity formulas.The similarity matrix is a m*m matrix with values calculated using the Cosine similarity. For example, matrix[i][j] stores the similarity between User i and User j. The higher the similarity is, the more similar the users are.

	User 1	User 2	...	User 400	User 401
User 1	0	0.38		0	0.42
User 2	0.22	0		0.32	0
...	...				
User 400	0.34	0.37	...	0	0.45
User 401	0.55	0	...	0.23	0

Tabe 4 similarity matrix(Cosine similarity formulas)

3. *The rating prediction*

The rating prediction is calculated by the formula shown below:

$$prediction(u,i) = \frac{\sum_{all\ movies\ rated\ by\ u\ \cap\ all\ movies\ be\ recommended\ to\ u}^{m} sim(i,m) \cdot R_m}{\sum_{all\ movies\ rated\ by\ u\ \cap\ all\ movies\ be\ recommended\ to\ u}^{m} |sim(i,m)|}$$

$$prediction(u,i) = \frac{sim(i,2) * 3 + sim(i,5) * 4 + sim(i,7) * 5 + sim(i,10) * 2}{|sim(i,2)| + |sim(i,5)| + |sim(i,7)| + |sim(i,10)|}$$

For example, movies ‘i’ has been recommended to user ‘u’, movies [2, 5, 7, 10] has been rated by the user ‘u’ and the rate for those movies are [3, 4, 5, 2].

Item-based Collaborative Filtering

Item-based collaborative filtering is a form of collaborative filtering for recommender systems based on the similarity between items calculated using people's ratings of those items. Item-based models use rating distributions per item instead of per user. When there are way more users than items, each item tends to have more ratings than each user, so an item's average rating usually doesn't change quickly. This leads to a more stable rating distributions in the model, so the model doesn't have to be rebuilt as often.

	Movie A	Movie B	Movie C
User A	5	/	5
User B	5	5	5
User C	5	/	Predicted rate = 5

Table 5 simple concept of item-based collaborative filtering

1. *The creation of the matrix storing the users' ratings for the movies*
   This is the same as user-based collaborative filtering.

2. *Item Similarity Computation*

	Movie1	…	Movie 'i'	...	Movie 'j'	…
User 1	4	…	4	...	5	...
User 2	3	…	4	...	3	...
User 'u'	/	...	3	...	5	...
…	...					
User 400	3	...	5	...	3	...
User 401	/	...	2	...	4	…

Table 6 item-based similarity computation

<u>Cosine Similarity</u>

$$sim(i,j) = cos(i,j) = \frac{\vec{i} \cdot \vec{j}}{\|\vec{i}\|_2 \cdot \|\vec{j}\|_2}$$

In table 5, User 1, user 2, user 'u', user 400 and user 401 watched both movie 'i' and movie 'j' and also rated these two movies. In the equation 1, i = [4, 4, 3, 5, 2] and j = [5, 3, 5, 3, 4].

<u>Jaccard Similarity</u>

$$sim(i,j) = \frac{|N(i) \cap N(j)|}{\sqrt{|N(i)||N(j)|}}$$

$$= \frac{\#movies\ have\ been\ rated\ by\ user\ i\ and\ user\ j}{\sqrt{(\#\ movies\ have\ been\ rated\ by\ user\ i)(\#\ movies\ have\ been\ rated\ by\ user\ j)}}$$

Jaccard index does not take the ratings into account, but only considers relative number of common items.

<u>Euclidean distance</u>

$$dis(i,j) = \sqrt{\sum (i_m - j_m)^2}$$

$$(where\ the\ i, j\ is\ the\ vector\ for\ movie\ i\ and\ movie\ j\ respectively\ )$$

$$sim(i,j) = \frac{1}{1 + dis(i,j)}$$

3. *Predicted Rate Computation*

Both user-based collaborative filtering and item-based collaborative filtering used weighted sum to calculate the predicted rate.

Results and Conclusion

1. Experimental Evaluation

1.1. Dataset

The major experimental data "ml-100k" was collected from the MovieLens web site (movielens.umn.edu) during the 17 months period from September 19th, 1997 through April 22nd, 1998. In this project, ml-100k consists of 100k ratings data from 943 users who rates at least 20 movies on 1682 movies. The ratings are integers from 1 to 5, with the greater score, indicating the higher preference of users on a movie. The dataset is divided into training set (u1.base) and test set (u1.test), 80% and 20%, respectively. Test set was used to evaluate the accuracy of the recommended set calculated by target users obtained by using the training set.

1.2. Evaluation Metrics

Since the absolute value function is not differentiable at all points, RMSE is more desirable as an objective function. Thus, RMSE was applied in this project, the formula is shown below:

$$Total\ Error\ = \sum_{u \in all\ users} RMSE_u$$

$$= \sum_{u \in all\ users} \sqrt{\frac{\sum_{i \in all\ movies\ user\ rated\ and\ be\ recommended} (r(u,i) - prediction(u,i))^2}{\#all\ movies\ user\ rated\ and\ be\ recommended}}$$

2. Experimental Procedure

In order to optimize the performance of the recommender system, both user-based and item-based collaborative filtering algorithm were tested and compared in the sensitivity of different parameters to determine the optimal values of theses parameters for the further experiment. The main related parameters are sensitivity of different similarity algorithms and sensitivity of different neighbourhood size which were only evaluated by training set.

3. Experimental Result

3.1. Effect of similarity algorithms

There are three different similarity algorithms which are Jaccard, Euclidean distance and Cosine similarity. Both two collaborative filtering algorithms were measured by using these similarity methods. And three different parameters of K(number of neighbours) are used for testing. As the following figures show, the results demonstrate that overall, Jaccard method has a clear advantage, as the RMSE maintains a lower level both in two CF algorithms comparing with those using other two methods. In addition, it lowers with the increase of the value of K. Since, Jaccard was selected as the optimal similarity method for the further experiment.

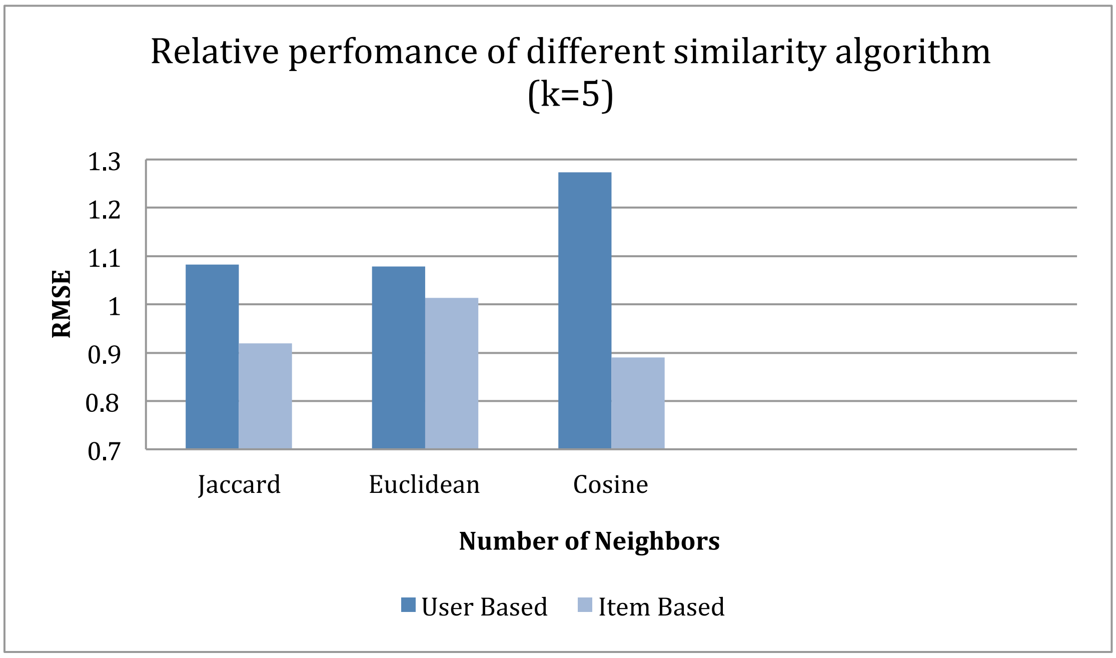

Figure1: Effect of the similarity computation measure on user-based and item-based collaborative filtering algorithm(k=5)

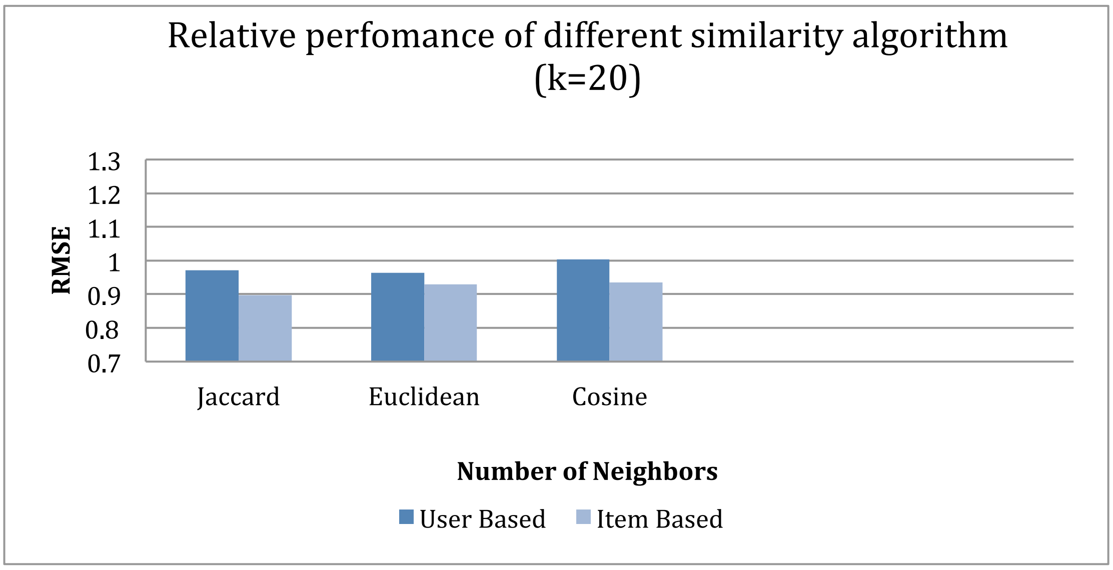

Figure2: Effect of the similarity computation measure on user-based and item-based collaborative filtering algorithm(k=20)

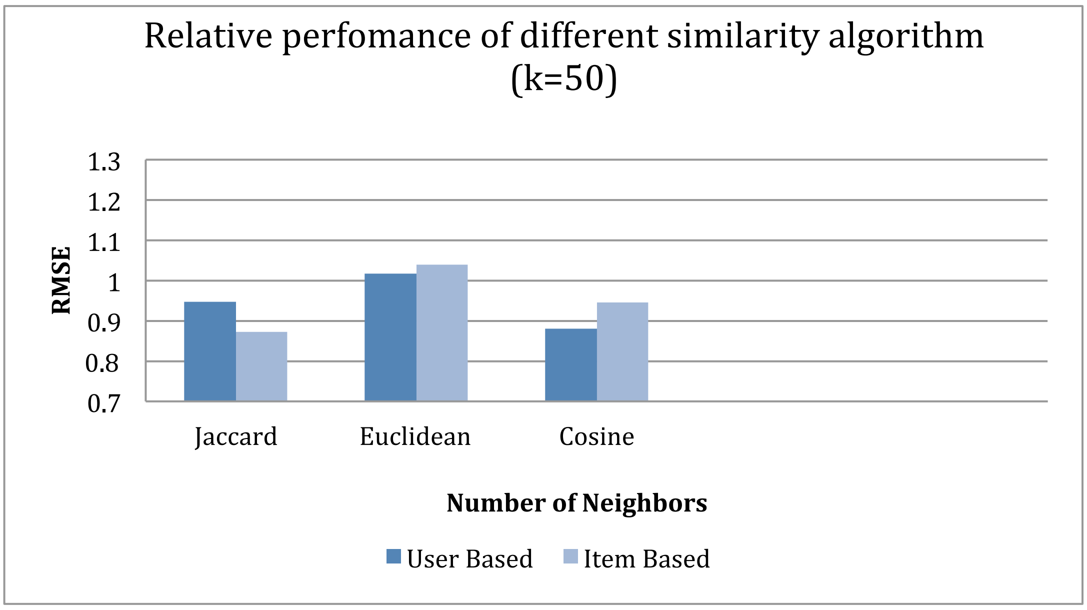

Figure3: Effect of the similarity computation measure on user-based and item-based collaborative filtering algorithm(k=50)

3.2. Effect of neighbour size

The size of the neighbour also has a significant impact on the prediction quality. This parameter was determined by computing the RMSE in both two collaborative filtering algorithms with Jaccard similarity methods for the clear observation. As the line chart reveals below, the RMSE of user-based model is the lowest if the value of K is 50, meanwhile the item-based model also has the lower level in the RMSE. Therefore, the value of K, which means the size of neighbourhood was assigned at 50.

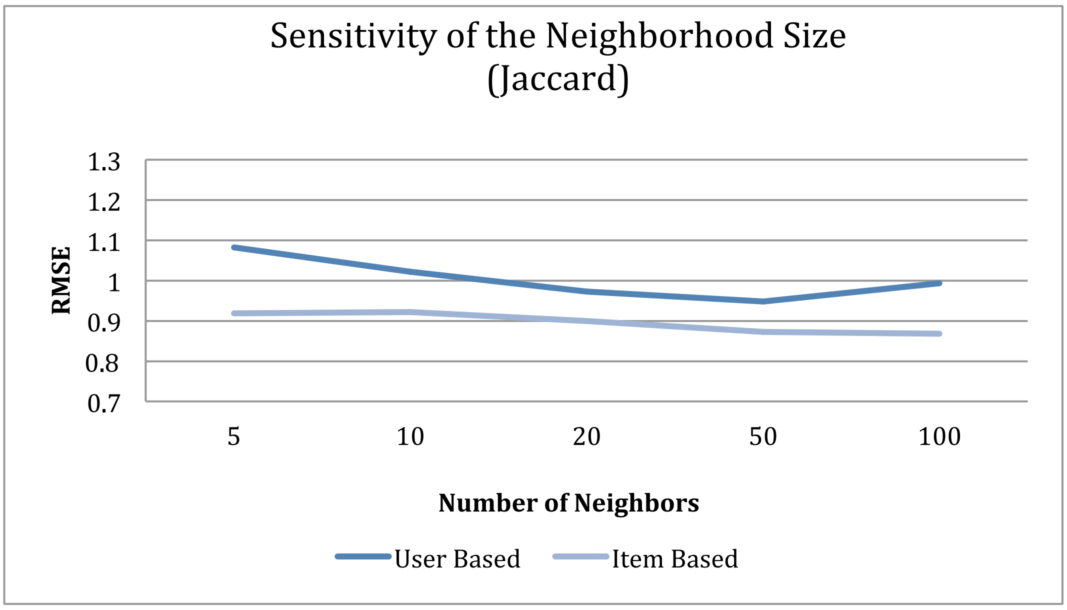

Figure4: Sensitivity of the parametre x on the neighborhood size (Jaccard)

3.3. Performance

Once we obtained the optimal values of the parameters, the performances of both collaborative filtering algorithms were compared and analysed. From the results, the item-based algorithm provides a better quality in general.

4. Conclusion

After comparing, it was concluded that item-based algorithm performed a better prediction in this case, in spite of a longer time taken in the code running. Item-based similarity was more static and provided the precomputation of the item neighbourhood which benefits the performance.

In real world, both algorithms of recommendation system are applied widely in different situations in our life based on their different characteristics.

The following table details the difference between two collaborative filtering algorithms in several features.

	User-Based CF	Item-Based CF
Scalability	Suitable for scenarios with a small number of users, if the number of users increases, it will be costly to calculate the similarity matrix between users.	Suitable for scenarios where the number of items is significantly smaller than the number of users. If there are many items, then it is costly to calculate the similarity matrix between items.
Domain	Applicable to real-time data and data with less personalized interest from users.	Applicable when there are many long-tailed products and users have strong personalized requirements.
Real-Time	A new behavior from user does not ensure an immediate change in the recommendation result.	A new behavior from user definitely leads to an immediate change in the recommendation result.
Cold Start	When a new user did some actions on a small number of items, he cannot get an updated recommendation immediately. This is because the user similarity table is usually calculated off-line at periodically. When a new item being added, once a user has acted on this item, the item can be recommended to other users similar to the user.	When a new user did an action on an item, he can get a recommendation if other items similar to this item. The item similarity table have to be updated(offline) before recommending new items to other users.

References:

[1] Sarwar, B., Karypis, G., Konstan, J. and Riedl, J., 2001, April. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th international conference on World Wide Web (pp. 285-295). ACM.
[2] Shani, G. and Gunawardana, A., 2011. Evaluating recommendation systems. In Recommender systems handbook (pp. 257-297). Springer, Boston, MA.
[3] Herlocker, J.L., Konstan, J.A., Terveen, L.G. and Riedl, J.T., 2004. Evaluating collaborative filtering recommender systems. ACM Transactions on Information Systems (TOIS), 22(1), pp.5-53.
[4] Zanker, M. and Jessenitschnig, M., 2009. Case-studies on exploiting explicit customer requirements in recommender systems. User Modeling and User-Adapted Interaction, 19(1-2), pp.133-166.
[5] Khoshneshin, M. and Street, W.N., 2010, September. Collaborative filtering via euclidean embedding. In Proceedings of the fourth ACM conference on Recommender systems (pp. 87-94). ACM.
[6] Ungar, L. H., and Foster, D. P., 1998 Clustering Methods for Collaborative Filtering. In Workshop on Recommender Systems at the 15th National Conference on Articial Intel ligence.
[7] Terveen, L., Hill, W., Amento, B., McDonald, D., and Creter, J., 1997. PHOAKS: A System for Sharing Recommendations. Communications of the ACM, 40(3). pp. 59-62.
[8] Shardanand, U., and Maes, P., 1995. Social Information Filtering: Algorithms for Automating 'Word of Mouth'. In Proceedings of CHI '95. Denver, CO.