Restaurant Recommender Systems with Collaborative Filtering

Yucian Hong

***Abstract*— This work researches recommender systems that recommend restaurants to target users. We experiment two approaches within the collaborative filtering algorithm, including memory-based and model-based collaborative filtering. Given that the experiment performs on a small data set of size O(1000) , the result implies that these algorithms may be viable for predicting user ratings and generate hits for user recommendations of restaurants. The result from model-based collaborative filtering is slightly better than collaborative filtering that only uses user similarities, but a bigger data size may yield more reliable results.**

Keywords—Recommender System, Collaborative Filtering

I. INTRODUCTION

Recommender systems are ubiquitous in predicting the interactions between businesses and customers. They are widely used by businesses to recommend products to users. They play critical roles in driving more revenues in all industries, including media, e-commerce, finance, advertisement, etc. Recommender models are effective in finding targeted content for targeted users, increasing the chance of hits in customer buys.

Recommender systems started with simple user clustering and category matching, and in the 1990s, collaborative filtering came into the scene. Collaborative filtering utilizes user and item metadata and their interaction score, to make predictions of users' preferences to particular items. Beyond collaborative filtering, there are also hybridized collaborative filtering adopting content-based approaches, as well as Bayesian inferences, etc. To date, collaborative filtering (CF) remains very effective and the most popular algorithm [1,2]. This paper will focus on collaborative filtering approaches in the study of recommending restaurants to users, using datasets that contains user profile and user rating. Section II discusses two CF algorithms, section III discusses the data, section IV shows the modeling result, and section V concludes this paper and discusses future improvement of this work.

II. RECOMMENDER SYSTEMS

Collaborative filtering makes use of user and item relations. Examples include user ratings on items, user-user similarities, item-item similarities[3], etc. The most popular memory-based models use user clustering to make recommendations. Model-based CF exploits user-item rating matrix to fill in the missing part of the matrix.

A. *Memory-based collaborative filtering*

This traditional CF algorithm calculates new user-item rating based on similarity between users [3,4]. User ratings are represented by an m by n matrix $A_{m \times n}$,

$$A_{m\times n} = \begin{bmatrix} R_{11} & R_{12} & \cdots & R_{1n} \\ R_{21} & R_{22} & \cdots & R_{2n} \\ \vdots & \vdots & & \vdots \\ R_{m1} & R_{m2} & & R_{mn} \end{bmatrix}$$

where m represents the number of users, and n represents the number of items. R_{ij} represents the rating of user i to item j. Each row in matrix A is the vector of a particular user, composed of its ratings on all n items [3]. The algorithm recommends particular items to a user based on the ratings of the most similar users. For example, the algorithm can recommend the top ranked items that have the most number of similar users, based on a set threshold of similarity score. Alternatively, it can also obtain an average rating among the top few similar users as the predicted rating of the new user, as done in this work. Common similarity scores include cosine similarity and Pearson correlation coefficient, etc. Cosine similarity score is the cosine of the two vectors $\mathbf{v_1}$, $\mathbf{v_2}$:

$$\text{Cosine similarity} = \text{Cos}(\mathbf{v_1}, \mathbf{v_2}) = \mathbf{v_1} \cdot \mathbf{v_2} / (|v_1||v_2|) \quad (1)$$

The time complexity of this algorithm is O(mn) in the worst case, for obtaining the prediction for one user-item pair. This poses major difficulties in computing. However, since the matrix is very sparse, that each user only rates a few items, only with a few users rating most items, the time complexity is more likely to be O(m+n). Though, this is still computationally intense when there are millions of users and products. For medium sized datasets, the traditional CF works effectively.

Besides crafting user vectors from their ratings on the items, one can also construct user vectors from their metadata, such as user location, gender, personality, dress preferences, etc. In this work, one of our model chooses a subset of user features to construct such vectors, and use cosine similarity to construct the cluster model.

B. *Model-based collaborative filtering*

Model-based CF algorithm does not rely on a set user-item matrix for making predictions, so it can alleviate the demanding time complexity in the traditional memory-based CF. The model learns the user-item interaction and makes its predictions. A popular algorithm of model-based CF is Singular Value Decomposition (SVD) [5]. From a "porous" user-item matrix that has missing rating data, SVD can learn how past user and item interact (ie. how user will rate the item), and predict the user-item interaction with a new user-item pair.

SVD decomposes the existing user-item matrix A into three matrices.

$$A = M\,D\,N^T \quad (2)$$

,where the left orthogonal matrix M contains information about the users, the right transposed orthogonal matrix N contains information about the items, and the D diagonal matrix contains the eigenvalues – the interaction factor between users and items. With the M, D, N matrices, when we want to find out the interaction score between user i and item j, we can simply take row i of M and column j of N^T, with the corresponding eigenvalue in D , to obtain R_{ij}. Therefore, this greatly reduces the time complexity compared to traditional CF.

III. DATA

The dataset used in this work is sourced from the UC Irvine Machine Learning Repository (https://archive-beta.ics.uci.edu/dataset/232/restaurant+consumer+data). The dataset contains user and restaurant metadata, such as user location, income level, age, restaurant cuisine type, as well as user ratings on restaurants. In this work, we leverage the user rating data and user metadata to construct our models. Since content-based collaborative filtering is outside of our scope, we do not use the restaurant metadata in our models. Future extension of this project can involve content-based CF and incorporate the full dataset.

A. *Data preview*

The user rating data contains 1161 data points. The dataset is shown as in figure 1. We utilize the userID, placeID, and rating columns to construct our CF models. The dataset does not contain missing or duplicate data, as inspected.

	userID	placeID	rating	food_rating	service_rating
0	U1077	135085	2	2	2
1	U1077	135038	2	2	1
2	U1077	132825	2	2	2
3	U1077	135060	1	2	2
4	U1068	135104	1	1	2
5	U1068	132740	0	0	0
6	U1068	132663	1	1	1
7	U1068	132732	0	0	0
8	U1068	132630	1	1	1
9	U1067	132584	2	2	2

Figure 1. Sample data of user (userID) rating on restaurants (PlaceID).

The user profile data contains user location, drink level, dress preference, transport preference, marital status, age, interest, personality, etc.

B. *Exploratory analysis*

The user ratings are on a scale of 0-2, and the distribution is as shown in figure 2. The class balance is good, so we don't need to worry about data imbalance.

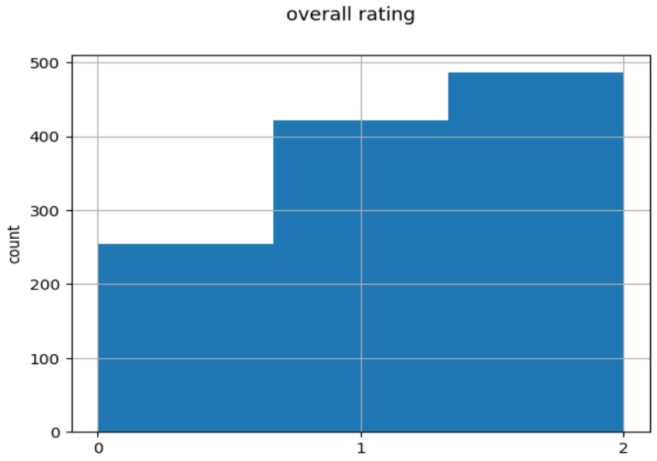

Figure 2. Histogram of user rating on restaurants.

Upon examination of user parameters and the distribution of their values (fig. 3), we found that not all parameters are useful in distinguishing between users. The user data is highly clustered at young, single, student groups that don't have dependents. Therefore, we exclude those data columns in our model.

```
activity
student           113
professional       15
?                   7
unemployed          2
working-class       1
Name: count, dtype: int64

marital_status
single     122
married     10
?            4
widow        2
Name: count, dtype: int64

budget
medium    91
low       35
?          7
high       5
Name: count, dtype: int64

hijos
independent    113
kids            11
?               11
dependent        3
Name: count, dtype: int64

dress_preference
no preference    53
formal           41
informal         35
?                 5
elegant           4
Name: count, dtype: int64
```

Figure 3. Distributions of values of a subset of user parameter (activity, marital status, budget, hijos, dress preference)

C. *Feature engineering*

For the memory-based model, we construct the user vectors by hand-picking informative features from the user profile. In addition, we transform categorical values into ordinal numbers. Scalable values are ordered in their numerical scale. We also choose to encode purely categorical values as numerical values instead of one-hot encoding them, in order to give each parameter equal weights. The transformed user features are as shown in figure 4.

	userID	birth_year	budget	dress_preference	interest	personality
0	U1001	2	1	0	4.0	3.0
1	U1002	2	0	0	3.0	2.0
2	U1003	2	0	2	1.0	1.0
3	U1004	0	1	0	4.0	1.0
4	U1005	2	1	1	1.0	3.0
...	...	...	...	...	...	...
133	U1134	2	1	1	4.0	1.0
134	U1135	1	0	0	4.0	2.0
135	U1136	2	0	1	2.0	3.0
136	U1137	2	0	2	0.0	1.0
137	U1138	2	1	2	4.0	3.0

138 rows × 6 columns

Figure 4. Transformed user features.

For the model-based CF, we simply adopt the user-item rating data frame.

IV. MODEL

Here we present the model evaluation metrics and modeling details.

A. *EVALUATION METRICS*

Common evaluation metrics for recommenders includes root mean square error (RMSE), mean absolute error (MAE), and fraction of concordant pairs (FCP) [6]. We incorporate RMSE and MAE in our work, and discuss their suitability.

1) *Root mean square error*

RMSE calculates the square-rooted average squared difference between the predicted and true ratings, which is formulated as:

$$\sqrt{\frac{1}{n}\sum_{u,i}(p_{u,i} - r_{u,i})^2} \tag{3}$$

, where n is the number of user, item pairs, u is the index of user, i the index of item, p the predicted rating, and r the true rating. It is useful when we want to penalize the larger errors (ie. outliers) more.

2) *Mean absolute error*

MAE calculates the average absolute difference between the true and predicted ratings, which is formulated as:

$$\frac{1}{n}\sum_{u,i}|p_{u,i} - r_{u,i}| \tag{4}$$

It does not exaggerate the error generated by outliers as RMSE, so is less prone to problems raised by those outliers. Both RMSE and MAE are suitable metrics for our work. Since the rating is on a very narrow scale of 0-2, and the rating counts are balanced, so there should not be bad outliers. MAE,

however, will be better than RMSE for the problem, because it gives equal weights to larger and smaller errors, and does not distort larger errors.

B. MODELING DETAILS

For the following model, we split the data into 7:3 train/test sets.

1) Memory-based CF

For the memory-based CF model, we compute the cosine similarity score between each pair of users. For predicting a particular user-restaurant rating on the test set, we rank the similarity score of users that rate the particular restaurant in the training set, and average the top 5 similar users' rating to be the predicted rating.

2) Model-based CF

We utilize the Surprise Python package [7] for constructing the SVD model. We adopt grid search for hyperparameter tuning. The hyperparameters are learning rate(0.001-0.01), regularization (0.01-10), the number of factors (2-100), forming a grid of 1404 sets of hyperparameters. (Note: previously we used a smaller random search grid of 500 on a different train-test split, and obtained slightly better results, but deleted it without backup.)

V. RESULT

Here we present the modeling result of our memory-based and model-based CF, evaluate the models, and discuss their implications.

A. Memory-based CF

For the memory-based CF model using user profiles as input vectors, we obtained MAE = 0.66, RMSE = 0.81 on the test set. Given the ratings are on a narrow scale of 0-2, the magnitudes of RMSE/MAE seem to indicate that the model doesn't work very precisely; however, the error isn't too outrageous (ie. a rating of 0 is likely to be predicted as 0.65-0.8, but not often over 1, so the prediction mostly stumble towards its nearest neighbor (0 → 1, 2→1), but not the next nearest neighbor(0→2)). The distribution of predicted and true ratings is shown in figure 5 below.

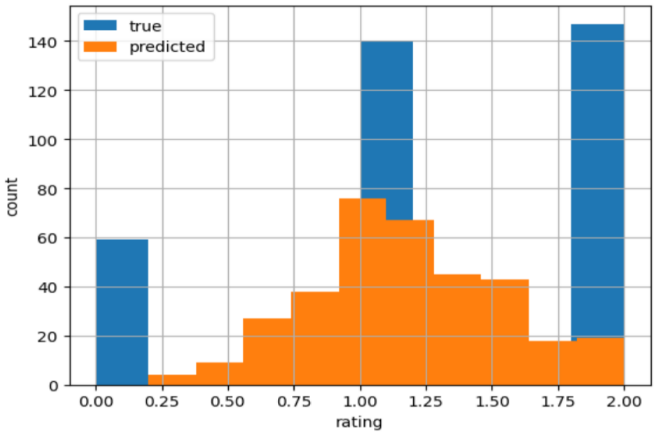

Figure 5. Histogram of predicted and true rating on the test set.

B. Model-based CF

Through the grid search, we found that the performance on test set is almost always with RMSE ~ 0.73 and MAE ~ 0.62, although the train set performance differs a lot. In a lot of the cases, the test error is much greater than train error, which could imply that those hyperparameters suffer from overfitting. Indeed, with the small data size of $O(10^3)$, the model can likely suffer overfitting. In the parameter sets that have similar RMSE/MAE for train and test, the error is slightly smaller than the memory-based collaborative filtering model. SVD is likely a better model than hand-picking similarity features, if given a better data size.

VI. CONCLUSION

The modeling results imply that collaborative filtering is a viable approach for recommending restaurants to target users, and that SVD may be a more powerful methodology than manually picking user features from user profiles, in addition to being computationally more efficient, which also enables a large scale grid search of hyperparameters. However, the limitation posed by a tiny set of data is severe, resulting in imprecise model results. In the memory-based model, it also suffers from insufficient user features, since the user demographics is not very broad. In addition to data size and feature space, we may also incorporate natural language processing techniques on restaurant review to obtain more informative data and further improve the model (ie. sentiment analysis, capture of categorized ratings) [2]. Besides the data collection, we may also achieve improvement by adopting content-based collaborative filtering by leveraging the existing restaurant metadata [3]. There are features with potentially strong connections between the users and restaurants, such as pricing and cuisine. Hyperparameter tuning also has space for improvement. Since the hyperparameters crosses several orders of magnitude, random search can be superior to grid search. In addition, FCP might be a more ideal metric for measuring the performance of the model.

REFERENCES

[1] J. L. Herlocker, J. A. Konstan, L. G. Terveen, J. T. Riedl, "Evaluating collaborative filtering recommender systems", ACM Transactions on Information Systems, Vol. 22, pp 5-, January 2004.

[2] R. M. Gomathi, P. Ajitha, G. H. S. Krishna and I. H. Pranay, "Restaurant Recommendation System for User Preference and Services Based on Rating and Amenities," *2019 International Conference on Computational Intelligence in Data Science (ICCIDS)*, Chennai, India, 2019, pp. 1-6, doi: 10.1109/ICCIDS.2019.8862048.

[3] G. Linden, B. Smith and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," in *IEEE Internet Computing*, vol. 7, no. 1, pp. 76-80, Jan.-Feb. 2003, doi: 10.1109/MIC.2003.1167344.

[4] A. Tripathi and A. K. Sharma, "Recommending Restaurants: A Collaborative Filtering Approach," *2020 8th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, Noida, India, 2020, pp. 1165-1169, doi: 10.1109/ICRITO48877.2020.9197946

[5] S. Funk, https://sifter.org/~simon/journal/20061211.html

[6] Al-Ghamdi, Maryam & Elazhary, Hanan & Mojahed, Aalaa. “Evaluation of Collaborative Filtering for Recommender Systems,” International Journal of Advanced Computer Science and Applications, vol. 12, Jan 2021, 12.10.14569/IJACSA.2021.0120367.

[7] Hug, N., (2020). Surprise: A Python library for recommender systems. Journal of Open Source Software, 5(52), 2174, https://doi.org/10.21105/joss.02174