A New Collaborative Filtering at Recommendation System Using Yahoo! Music Data

Kim Yejin
School of Management Engineering
UNIST
Kimyejin99@unist.ac.kr

Lee Yeongho
School of Management Engineering
UNIST
lyh1030@unist.ac.kr

Abstract
The explosive growth of the world-wide-web and the emergence of e-commerce has led to the development of recommender systems. Recommender systems are personalized information filtering used to identify a set of items that will be of interest to a certain user. Despite the usefulness of the CF method for successful recommendation, several limitations remain, such as sparsity and cold-start problems that degrade the performance of CF systems in practice. Our work proposes a comprehensive collaborative filtering model. In addition to using the CBF method-cosine similarity, the proposed model considers the relationships of the music genre. Our model is to generate the matrix of genre-music while maintaining user's favorites by SVD of genre Similarity matrix and predicting unobserved rating quickly while maintaining original property.

I. Introduction

The recommendation system is part of routine life where people rely on knowledge for deciding their interests. However, due to the variety of items and lack of prior knowledge, there are difficulties in purchasing what they want, and from the seller's perspective, recommending the appropriate items and linking them to purchasing is directly related to profit generation. The recommendation system is defined as the supporting system which is used to help people to find appropriate services, or products by analyzing the suggestions from other users, that reviews from other authorities and user attributes. It provides the personalized recommendation services and contents to the different users.[5]

Collaborative Filtering is one of the most widely used and successful technologies in Recommendation system.[6][7] Collaborative Filtering-based recommendation techniques have achieved great success and have a wide range of application prospects in many fields. The Collaborative Filtering makes recommendations for the current active user using users' historical rating information without analyzing the content of the information resource. However, in recent years, data sparsity and high dimensionality brought by big data have negatively affected the efficiency of the traditional Collaborative Filtering recommendation approaches.[8] So, in Collaborative Filtering, the context information (all categories that represent the setting in which recommendation is deployed) is added into Recommendation System to construct a training model to further improve the recommendation accuracy and user's satisfaction. Then, various kinds of hybrid Collaborative Filtering algorithms have emerged.

In this paper, we suggest a novel method of applying music-metadata to well-known item-based CF methods, which effectively employs metadata for calculating inter-similarities among music-albums and build a user-genre matrix. Based on this matrix, we propose content-metadata-based approaches. Our approach significantly use CBF method (cosine Similarity) and also provides effective recommendations in a large-scale data.

II. Related Work

A complete Recommendation System consists of the three parts: user, item resource and algorithm.[9][10] The user model is established by analyzing the users' interests and preferences, likewise, the model for item resource is established according to items' feature. Then, the characteristics of the user are compared with the characteristics of all items to predict which items the user might like by using the recommendation algorithm, and the predicted results are recommended to the user. Mainly the Recommendation Methods are classified into three categories:

A. Contents-Based Filtering

This system recommends items based on product description or content of items rather than other users' ratings of the system. This system uses the item-to-item correlation rather than user-to-item correlation for creating recommendation. In this system the recommendation process first starts by gathering data or information about the items (author, title, cost, synopsis, production year, etc.). Most of this type of system use feature extraction techniques and information indexing to extract the content data.

In content-based filtering, this system processes information and data from various sources and try to extract useful features and element about the contents of the items. In this system the constraint-based filtering uses features of items to determine their relevance. This approach does not require data of other users and it has capability of recommending item to users with unique taste and does not suffer from problems. This method is widely used to recommend text-based news or Internet articles, as well as movies, music and books, due to its ease of analysis and the availability of metadata.

B. Collaborative Filtering

Collaborative filtering is a method of predicting preferences based on the basic assumption that "customers with similar preferences for a particular item will show similar preferences for other items." Content-based approaches rely solely on user and item information The biggest difference is that content-based approaches rely on user and item information to predict preferences, while collaborative filtering uses information evaluated by users on items to predict preferences. In other words, it can ensure the diversity of recommended items because it selects customers with similar tastes and recommends their preferred items to the recommended customers.

The basic concepts are as follows. When a recommended customer is selected, the similarity between the recommended customer and other users is measured based on the historical data. That is, the more items purchased match, the more similar tastes they have, the higher similarity they have, and when they measure similarity with all users, the most similar user is chosen as their neighbor.

Such Collaborative Filtering is largely divided into memory-based Collaborative Filtering and model-based Collaborative Filtering. Memory-based Collaborative Filtering is a method described earlier that calculates similarity between users and recommends items selected by users with high similarity. Model-based Collaborative Filtering is based on the process of memory-based Collaborative Filtering methods but utilizes machine learning or data mining techniques in the stages of clustering, classification, and prediction.

1. *Memory-based Collaborative Filtering*

Memory-based Collaborative Filtering recommendation algorithm obtains the similar relationships between users or items according to the user-item rating matrix, and then recommends the items that are highly rated by similar users for the active user. In memory-based Collaborative Filtering, the ratings on items from users are directly used to predict unknown ratings for new items. The memory-based recommendation method can be subdivided into two ways: user-based CF and item-based CF. [11]

(1) User-based Collaborative Filtering

The idea of user-based CF is that users with similar historical ratings should have similar interests, so we can predict the active user's missing ratings on the specific items according to similar users' ratings on given items. Firstly, the similarities between the active user and other users are calculated, and then the neighbors of the active user are selected according to the similarities. Finally, the ratings from the active user are predicted according to the historical preference information of the similar neighbor users, and the recommendation results are generated.[12]

(2) Item-based Collaborative Filtering

Similar to the user-based Collaborative Filtering algorithm, the item-based Collaborative Filtering algorithm is also executed in the following three steps: (1) Calculate the similarity between items according to the user-item rating matrix; (2) Select the similar neighbor items according to the similarity; (3) Predict unknown ratings on the active item according to the neighbor items, and generate a recommended list.

2. *Model-based Collaborative Filtering*

The memory-based Recommendation System is simple to implement, and the algorithms is easy to understand. However, the memory-based Recommendation System is not suitable for practical applications when dealing with large amounts of users and items. In this case, the model-based Recommendation System emerge subsequently, which can avoid that issue. The model-based Recommendation System requires a learning phase in advance for finding out the optimal model parameters before making a recommendation. Once the learning phase is finished, the model-based Recommendation System can predict the ratings of users very quickly. Among them, latent factor model (LFM) is very competitive and widely adopted to implement Recommendation System, which factorizes the user-item rating matrix into two low-rank matrices: the user feature and item feature matrices. It can alleviate data sparsity using dimensionality reduction techniques and usually produce more accurate recommendations than the memory-based CF approach, while drastically decreases the memory requirement and computation complexity.[13][14][15][16][17][18]

C. *Hybrid Filtering*

In order to provide users with satisfactory recommendation results, they should be aware of the user's current situation, accurately understand the characteristics of their preferences, and recommend items appropriate for them. To solve the problems of scalability, cold start, and data sparse, many traditional technologies are combined with each other, such as the time context, and trust relationship between users are integrated into Recommendation Systems.
In some cases, you can find a movie that has a similar atmosphere to the movie you have seen so far, or you can find another movie with an actor who has appeared in the movie you have watched so far. In other words, the algorithms used in recommendation systems have different criteria for recommendation, so we need to select the appropriate algorithms for the characteristics of the field in which the recommendation system is applied and use as much information as possible to improve recommendation performance.

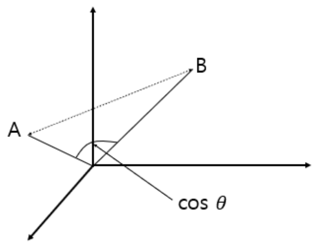
As previously seen, content-based approaches and collaborative filtering methods have their own advantages and disadvantages. Content-based approaches can recommend items that have not been evaluated but tend to be over-specialized. On the contrary, collaborative filtering shows high accuracy, but is not recommended for items that have not been evaluated by users. Therefore, research on hybrid recommendation systems has recently become important, which can compensate for shortcomings and effectively utilize various information while maximizing the advantages of each method.

III. Literature Review

A. *Cosine Similarity [24]*

The collaborative filtering algorithm use the matrix that represent a client as N-dimensional vector of items. Each vector has the implicit or explicit data like purchased or rating. Using this data, Cosine similarity can compute the similarity between two N-dimensional vectors based on their angle. It can compute the similarity between user and user, or item and item, or user and item. So the value obtained by the calculation is called similarity scores. These scores are the basis for the recommendation system based on users or items. The formula of cosine similarity is as follow:

$$\cos similarity\,(A, B) = \frac{A \cdot B}{\|A\| \times \|B\|}$$



B. Matrix Factorization Techniques for Recommender Systems [19]

In 2006, Netflix began their Netflix Prize contest to find a more efficient and accurate movie recommendation system to replace their current system. They promised a prize of one million dollars to anyone who could improve over the 'Cinematic' system by at least 10% Root Mean Squared error (RSME). After many years of the contest, prize was awarded to team and people found out the winning entry of Netflix Prize used a number of SVD(Singular Value Decompositions) models. After that SVD(Singular Value Decompositions) have become very popular in the field of Collaborative Filtering. Since SVD can be applicable for all $m \times n$ matrices, it allows to effectively handle collaborative filtering methods that increase the scarcity and decrease performance as the number of users increases.

C. MMCF: Multimodal Collaborative Filtering for Automatic Playlist Continuation [20]

In this paper, they propose a multimodal collaborative filtering model to deal effectively with diverse data. This consists of two components: (1) an autoencoder using both the playlist and its categorical contents and (2) a character-level convolutional neural network using the playlist title only. By simultaneously analyzing the playlist and the categorical contents, their model successfully addresses the cold-start and popularity bias problems. In addition, they consider the context of a playlist by utilizing its title, thus enhancing the prediction of well-suited tracks.

D. Boosting Memory-Based Collaborative Filtering Using Content-Metadata [21]

Many recommendation systems employ the memory-based collaborative filtering (CF) method, which has been generally accepted as one of consensus approaches. Despite the usefulness of the CF method for successful recommendation, several limitations remain, such as sparsity and cold-start problems that degrade the performance of CF systems in practice. To overcome these limitations, they propose a content-metadata-based approach that uses content-metadata in an effective way. By complementarily combining content-metadata with conventional user-content ratings and trust network information, our proposed approach remarkably increases the amount of suggested content and accurately recommends a large number of additional content items.

IIII. Method

A. K-NN (K-Nearest Neighbor)

The K-Nearest Neighbor (K-NN) algorithm is one of the simplest methods for solving classification problems. It clustered the data according to the distance. User-based collaborative filtering has the information of each user and using this we can calculate the distance between users. So, it find other users that have similar interest, that means have largest similarity. Recommended to target user is then generated based on K-NN method. The process of recommendation using K-NN is this. First, calculate the similarity between user and user using the formula. Formula can be cosine similarity, Euclidean similarity, etc. after that select the highest similarity with the target user's K-Nearest users as nearest neighbor set.

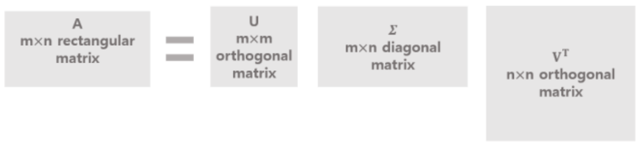
B. SVD (Singular Value Decompositions)

SVD is a method of making recommendations based on the global structure of matrices. It is a kind of dimension reduction and can reduce high-dimensional matrices to low-dimensional matrices to increase the accuracy of analysis and speed up computation. SVD decomposes the original matrix into two matrices and one diagonal matrix. In the recommendation system, dimensionality reduction can be performed using each decomposed matrix. The value of decomposition is useful because it is applicable for all m × n matrices, regardless of whether the matrix is a square or not. Although there are many models that decompose the matrix, SVD is known to define the latent factor well when the matrix is decomposed. In particular, it is used when there is a lot of missing data about the users and items. SGD (Stochastic Gradient Descent) methods are used for learning feature matrices by constructing arbitrary feature matrices to minimize the value of difference between observed target matrices using gradient descent techniques. In addition, ALS(Alternating Least Squares) techniques for anchoring one of the arbitrary feature matrices and alternating learning Latent Factor once in a while are also widely used. The ability to reduce dimensionality using SVD enables fast calculation and real-time recommendation.

Using SVD, we can decompose matrix A containing user and item data into matrix $U,\ \Sigma,\ V$ So We can express matrix A as below expression:

$$A = U\Sigma V^T$$

To explain each matrix, U is the eigenvalue decomposition of AA^T, and V is transpose of eigenvalue decomposition of $A^T A$. And Σ is diagonal matrix, and diagonal is consisted of $\sqrt{\lambda_1}, \sqrt{\lambda_2}, \sqrt{\lambda_3}$.



Gradient descent is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. The object of the gradient descent is minimizing the loss function, and loss function is the difference between value that we get and real value. It iterates to direct opposite side of gradient to achieve it object. Batch Gradient Descent use whole train set when we calculate loss function. But if we do this, it need many calculations and take lots of time. To prevent this, normally Stochastic Gradient Descent is used. It uses only part of the data so it can reduce amount of calculation and time. So, at first, SGD randomly select U, V^T matrices and calculate A. and then compare this estimated matrix with real matrix. After calculating the error between estimated matrix and real matrix, update estimated matrix with L2 regulation to minimize the error. The formula of updating the matrices using SGD is as follow:

$$error = (A_{real} - A_{estimated})^2$$
$$U_{new} = U - \eta(error * V + L_2 * U)$$
$$V_{new} = V - \eta(error * U + L_2 * V)$$

As repeat this until having smallest error, we can get the estimated data.

Alternating Least Squares fixed one of the matrices and alternating learning Latent Factor repeatably. It fix item matrix as constant and train the user matrix, and then it fix user matrix as constant and train item matrix. As repeat this process, it learns best user and item Latent Factor. But in this ALS algorithm, when it updates loss function, it have to consider the presence of data. If there is the data, we have to consider that. So put 1 to it. And if there is not data, we have to exclude it. So put 0 to it. Therefore, the formula of updating the matrices using ALS is as follow:

$$p \begin{cases} 1 \; if \; u > 0 \\ 0 \; if \; u = 0 \end{cases}$$
$$U_{new} = (V^T V + \lambda I)^{-1} V^T P$$
$$V_{new} = (U^T U + \lambda I)^{-1} U^T P$$

C. Clustering for Unknown Music

1. UMAP(Uniform Manifold Approximation and Projection for Dimension Reduction)[22]

UMAP (Uniform Manifold Approximation and Projection) is a novel manifold learning technique for dimension reduction. UMAP is constructed from a theoretical framework based in Riemannian geometry and algebraic topology. The result is a practical scalable algorithm that applies to real world data. The UMAP algorithm is competitive with t-SNE for visualization quality, and arguably preserves more of the global structure with superior run time performance. Furthermore, UMAP has no computational restrictions on embedding dimension, making it viable as a general-purpose dimension reduction technique for machine learning.

First, we make unknow music-user matrix. And then, make album-user matrix. To get user's similarity on genre, we calculate cosine-similarity of album. Using that cosine-similarity, since cosine-similarity matrix have huge dimension, we use UMAP. After that, we use cluster method - HDBSCAN.

2. HDBSCAN (A Hybrid Approach To Hierarchical Density-based cluster Selection) [23]

HDBSCAN is a density-based clustering algorithm that constructs a cluster hierarchy tree and then uses a specific stability measure to extract flat clusters from the tree. We show how the application of an additional threshold value can result in a combination of DBSCAN* and HDBSCAN clusters and demonstrate potential benefits of this hybrid approach when clustering data of variable densities. Our approach is useful in scenarios where we require a low minimum cluster size but want to avoid an abundance of micro-clusters in high-density regions. The method can directly be applied to HDBSCAN's tree of cluster candidates and does not require any modifications to the hierarchy itself. It can easily be integrated as an addition to existing HDBSCAN implementations.

After using UMAP, we apply HDBSCAN, then we can get each cluster at unknown genre.

V. Application

A. About our dataset

This dataset represents a snapshot of the Yahoo! Music community's preferences for various songs. The dataset contains over 717million ratings of 136 thousand songs given by 1.8 million users of Yahoo! Music services. The data was collected between 2002 and 2006. Each song in the data set is accompanied by artist, album, and genre attributes. The users, songs, artists, and albums are represented by randomly assigned numeric id's so that no identifying information is revealed. The mapping from genre ids to genre, as well as the genre hierarchy, is given. This dataset has 85% of unknown music.

Standard pre-processing for collaborative filtering and rating prediction was applied to create this data set. The data has been trimmed so that each user has rated at least 20 songs, and each song has been rated by at least 20 users. The data has been randomly partitioned into 10 equally sized sets of users to enable cross-validation techniques. The available ratings for each user have been randomly partitioned into training and test sets to enable all-but-K testing protocols. The test set consist of 10 ratings, and the training set consists of the remaining ratings.

B. about our model

This study is conducted to make the recommendation system more efficient with the song data mentioned above. We were focused on the reducing the time that takes for the algorithm to run when the new user come in. In addition, we thought of ways to reflect information such as album and singer of song, different from the existing SVD algorithm that used only users, songs, and ratings. The algorithm shows in the following steps:

Step 1: Prepare ‘Yahoo! Music User Ratings’ data and merge each data set.

Step 2: Get Cosine-Similarity of Album to Cluster ‘Unknown’ genre

Step 3: Use UMAP as pre-processing to boost HDBSCAN.

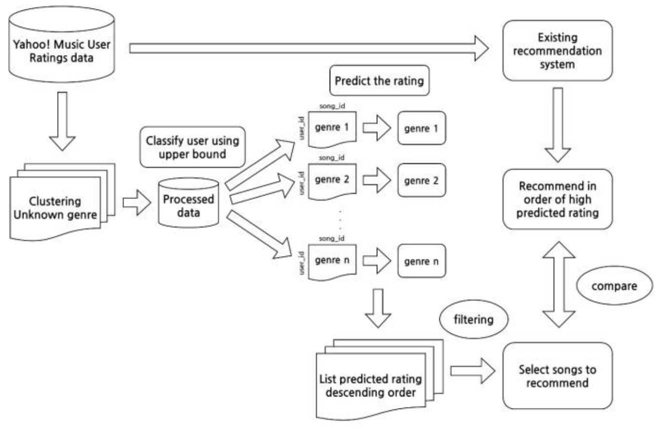
Step 4: Cluster ‘Unknown’ genre using HDBSCAN.

Step 5: Classify user into each genre category using upper bound formula.

Step 6: Run the SVD algorithm for each genre category to predict the rating.

Step 7: List predicted rating descending order.

Step 8: select songs to recommend.


Yahoo! Music User Ratings data
Existing recommendation system
Predict the rating
Classify user using upper bound
Clustering Unknown genre
Processed data
song_id
user_id
genre 1
genre 2
genre n
Recommend in order of high predicted rating
compare
filtering
List predicted rating descending order
Select songs to recommend

C. Process of implement

Before using model, preprocess of three data sets that train data, song attribute, genre. Merge the given datasets into one dataset. We will use meta data(genre)of the song. Next, preprocess about ‘Unknown genre’ music. Datasets have 85% of ‘unknown genre’ This is inappropriate to use our proposed model because the unknown genre occupies a very large portion. So, we clustered the unknown genre in three ways: After clustering, one cluster was treated as a genre.

First, use cosine- similarity for each album. We gave similarities according to the tendency of users. Currently, each score of albums was below.

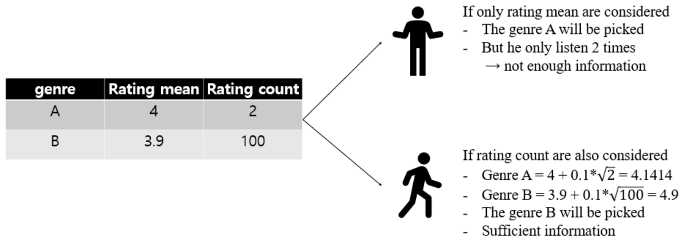
$$Score = rating\ mean + 0.5 * \sqrt{rating\ count}$$

because it is reasonable to use both the rating score and the count of rating. We get similarity matrix of each album. Next, we used UMAP to perform non-linear manifold aware dimension reduction. So, we could get the dataset down to a few dimensions small enough for a HDBSCAN. Finally, we used HDBSCAN for clustering Unknown Genre Music.

After clustering Unknown genre, the processed data can be obtained. And then the algorithm classify user using upper bound. The process starts with make the dataframe with ‘user_id’, ‘song_id’, ‘rating’, ‘category’. And then calculate the average score of each category by each user. Each user also makes a data frame about how many times they have listened to that category's song. Then combine the two dataframe. Finally, for each category, apply this average rating and the number of hearing to the upper bound expression to give the user the best song category. The upper bound expression is as follow:

$$\text{Score} = \text{rating mean} + 0.1 * \sqrt{\text{rating count}}$$

The reason that why using upper bound expression is that we could give more suitable genre category to users by considering the rating count. It will be more understandable if you think about the following examples. Let's say that a user gave a rating of 4 and 3.9 for genre a and b. In fact, there is not much difference between 4 and 3.9. But means are the only reason why Genre a is chosen. In fact, the user only listened to two genre A songs. Therefore, genre b seems more appropriate to the user. Therefore, we used the upper bound formula to weight the genre that user heard more about to prevent this problem.

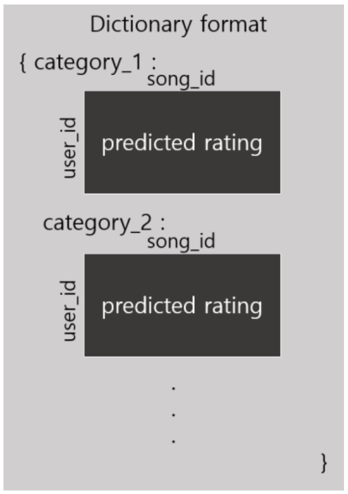


genre	Rating mean	Rating count
A	4	2
B	3.9	100

If only rating mean are considered
- The genre A will be picked
- But he only listen 2 times
  → not enough information

If rating count are also considered
- Genre A = 4 + 0.1*$\sqrt{2}$ = 4.1414
- Genre B = 3.9 + 0.1*$\sqrt{100}$ = 4.9
- The genre B will be picked
- Sufficient information

Through the previous process, algorithm gave users the best category. Therefore, each category gathered data about user_id and the song that user heard. Then algorithm run the recommendation system algorithm with the data. Each category name and dataframe with a recommended algorithm were saved as key and value in a dictionary, respectively.



The algorithm we used to run the recommended system is SVD. The algorithm was expressed by python coding, so we can find a way to implement. Using these coding implementations, we can take out the matrix directly when it needed. After specifying learning rate, regulation param, and number of iterations in class, algorithm can predict the rating of songs through the SGD algorithm.

After running SVD algorithm, empty space in the matrix was filled with predicted rating. So according to this, algorithm can arrange predicted rating descending order and select songs to recommend. So first, arrange the predicted rating in order of large values. Choose the top 100 songs in order. The new data frame consists of index as user_id, columns as rating, and value as song_id. Nan value is the value that occurs when there are less than 100 songs in each category.

VI. Results and Discussion

We try to achieve various results through our project. First of all, we compare the running time between original SVD algorithm and the algorithms we made. Using the time module within Python, the algorithm could be determined how long it would take to run. The results are as follows:

	Computing Time
Original SGD	590sec
Proposed model	60sec

While original SVD takes 590 minutes, we can see that our algorithm takes only 60 minutes. Also, we assume the situation that new user come and express time complexity. Computing the SVD of and $m \times n$ matrix has complexity as follow:

$$O(mn\,min(n, m))$$

We can see it becomes computationally expensive for large data sets. But our proposed model used only the category data in which target user belong. So based on the assumption that all categories have same number of songs and put the number of categories as k, time complexity of our proposed model is as follow:

$$O(m\frac{n}{k}\ min(\frac{n}{k}, m)).$$

Considering the situation in which new users come in, original SVD takes the same amount of time because it needs to rerun using all of the data, but our algorithm can reduce the time even more because new user can be applied in most appropriate category after identifying the taste and then we just need to run the algorithm of that category. In reality, new songs and new users come in so actively, so if we use our algorithm, we can present the recommended song more quickly to users.

VII. Conclusion

A recommendation system refers to a system that provides recommendation by information filtering about specific items that are appropriate for the user among several. The word The term 'filtering' here refers to the technology of choosing the appropriate item among several items. As technology advances further, the importance of recommendation is emerging, and research is actively underway on various techniques to recommend appropriate products.

In this paper, we explore the features of various recommendation systems such as contents-based filtering, collaborative filtering, and hybrid filtering. Among them, we could learn more about collaborative filtering, a way to recommend music to users through Yahoo music data. We also mathematically understood the K-NN and SVD algorithms which is used in the original collaborative filtering. Then, we designed an algorithm that can be recommended by adding new information and reducing the time complexity that away from existing user, item, and rating matrix recommendation method. Furthermore, we proceed with clustering using UMAP and HDBSCAN to more effectively apply category data provided by Yahoo music data to algorithms. We then apply the SVD algorithm for each category to predict ratings and implement an algorithm that recommends music to users based on it. Comparing the computation time with existing algorithms was able to conclude that our algorithm would be suitable for real-world situations where new users and music continue to emerge.

VIII. Future Work

Despite the results of the study, there are some limitations. First, the actual data used after preprocessing is 10,000, which can be said to be a rather insufficient sample to yield stable collaborative filtering results. Second, the optimal combination of the number of users and the number of clusters utilized in the expected evaluations experimentally obtained through Cosine-Similarity, UMAP and HDBSCAN varies as the size and group of samples change. The data analysis of this study shows that the performance of recommendation systems is affected by the size of Matrix. Third, we verified the performance of our model by comparing with existing algorithms and computing time calculations, but a new user and new music is required for more rigorous verification. Therefore, if the data is collected later and compared to the actual score and calculation time after the music recommendation of the new customer, we will enable rigorous verification.

Although various influence factors are considered to improve the performance of Recommender System, it will increase the parameter setting and time complexity of the model. In addition, it is difficult to obtain the optimal value due to too many parameters. With the development of deep learning technology, deep learning has gradually been applied in Recommender System due to strong feature representation, and it can learn the latent item association from the user-item rating directly for predictive recommendation without employing a similarity measure. Therefore, in the future research of Recommender System, to achieve better performance, we should focus on how to use deep learning technology to solve the problems of data sparsity, cold start and complexity.

References

[1] Rofeca, G. R., & Js, S. (2017). Book recommendation using cosine similarity. International Journal of Advanced Research in Computer Science, 8(3)

[2] Gower, S. (2014). Netflix prize and SVD.

[3] Bennett, James, and Stan Lanning. "The netflix prize." Proceedings of KDD cup and workshop. Vol. 2007. 2007.

[4] Vasudevan, Vinita, and M. Ramakrishna. "A hierarchical singular value decomposition algorithm for low rank matrices." arXiv preprint arXiv:1710.02812 (2017).

[5] Wang, L. C., X. W. Meng, and Y. J. Zhang. "A heuristic approach to social network-based and context-aware mobile services recommendation." Journal of Convergence Information Technology 6.10 (2011): 339-346.

[6] Resnick, Paul, et al. "Grouplens: An open architecture for collaborative filtering of netnews." Proceedings of the 1994 ACM conference on Computer supported cooperative work. 1994.

[7] Melville, Prem, Raymond J. Mooney, and Ramadass Nagarajan. "Content-boosted collaborative filtering for improved recommendations." Aaai/iaai 23 (2002): 187-192.

[8] Gong, Songjie. "A collaborative filtering recommendation algorithm based on user clustering and item clustering." JSW 5.7 (2010): 745-752.

[9] Zhang, Z., and H. Liu. "Research on personalized recommendation models and algorithm in social networks." Shandong Normal University (2015).

[10] Ren, Lei. "Research on some key issues of recommender systems." East China Normal University (2012).

[11] Yang, Zhe, et al. "A survey of collaborative filtering-based recommender systems for mobile internet applications." IEEE Access 4 (2016): 3273-3287.

[12] Adomavicius, Gediminas, and Alexander Tuzhilin. "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions." IEEE transactions on knowledge and data engineering 17.6 (2005): 734-749.

[13] Lü, Linyuan, et al. "Recommender systems." Physics reports 519.1 (2012): 1-49.

[14] Zhou, Xun, et al. "SVD-based incremental approaches for recommender systems." Journal of Computer and System Sciences 81.4 (2015): 717-733.

[15] Sarwar, Badrul, et al. Application of dimensionality reduction in recommender system-a case study. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[16] Sherif, Nurudeen, and Gongxuan Zhang. "Collaborative filtering using probabilistic matrix factorization and a Bayesian nonparametric model." 2017 IEEE 2nd International Conference on Big Data Analysis (ICBDA). IEEE, 2017.

[17] Huang, Shanshan, et al. "A hybrid multigroup coclustering recommendation framework based on information fusion." ACM Transactions on Intelligent Systems and Technology (TIST) 6.2 (2015): 1-22.

[18] Mnih, Andriy, and Russ R. Salakhutdinov. "Probabilistic matrix factorization." Advances in neural information processing systems 20 (2007): 1257-1264.

[19] Koren, Yehuda, Robert Bell, and Chris Volinsky. "Matrix factorization techniques for recommender systems." Computer 42.8 (2009): 30-37.

[20] Yang, Hojin, et al. "Mmcf: Multimodal collaborative filtering for automatic playlist continuation." Proceedings of the ACM Recommender Systems Challenge 2018. 2018. 1-6.

[21] Kim, Kyung Soo, Doo Soo Chang, and Yong Suk Choi. "Boosting Memory-Based Collaborative Filtering Using Content-Metadata." Symmetry 11.4 (2019): 561.

[22] McInnes, Leland, John Healy, and James Melville. "Umap: Uniform manifold approximation and projection for dimension reduction." arXiv preprint arXiv:1802.03426 (2018).

[23] Malzer, Claudia, and Marcus Baum. "A Hybrid Approach To Hierarchical Density-based Cluster Selection." 2020 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI). IEEE, 2020.

[24] Asanov, Daniar. "Algorithms and methods in recommender systems." *Berlin Institute of Technology, Berlin, Germany* (2011).