EIE3280 Final Report: Comparative Anime Recommendation System Exploration with DualTaste Recommender

Dongzhuyuan Lu
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Email: 119010216@link.cuhk.edu.cn

Junhao Ran
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Email: 119010252@link.cuhk.edu.cn

Taimeng Fu
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Email: 119010073@link.cuhk.edu.cn

Yifan Wu
School of Data Science
The Chinese University of Hong Kong, Shenzhen
Email: 119010341@link.cuhk.edu.cn

Yabin Cheng
School of Science and Engineering
The Chinese University of Hong Kong, Shenzhen
Email: 119010046@link.cuhk.edu.cn

***Abstract*—In this work, we developed a hybrid recommendation system for evaluating anime films, which combines collaborative filtering and content-based filtering methods. For collaborative filtering, we apply two existing methods to a new application scenario (Kaggle Anime Recommendation Database) to verify the performance and accuracy. Next, we propose a new content-based method *DTAR*, specifically targeting on this dataset, aiming to solve challenges, including complicated data composition, cold-start problem, and so on. We conduct several experiments on the dataset and found that our proposed *DTAR* method does not differ by the two state-of-the-art algorithms, while solving the above-mentioned issues.**

I. Introduction

Recommendation systems play a crucial role in suggesting interest-based contents to users, especially on mobile content platforms where users rely heavily on recommendations rather than manual search results. It is generally accepted that the quality of recommendations significantly affects the user experience. As reported by [1], the popularity of anime in China has been steadily growing at an annual rate of 12 to 25 percent since 2017. However, compared to short-form videos on platforms like TikTok and Instagram, anime often lacks sufficient user preference data, making it challenging to build an effective recommender.

Our research aims to address this challenge by building two separate recommendation systems using content-based filtering and collaborative filtering techniques.

In terms of Collaborative Filtering algorithms, a recommendation system was developed leveraging the robust capabilities of the Neural Collaborative Filtering (NCF) methodology. NCF, an innovative approach to recommendation systems, capitalizes on the power of deep neural networks to extract user-item interaction patterns, providing a more nuanced appreciation of these interactions than traditional matrix factorization methods [4]. The model's performance demonstrated the efficiency and effectiveness of NCF in capturing the intricacies of user-item interactions and highlighting its superiority over traditional methods. Furthermore, the project underlines the potential of AI in revolutionizing content delivery systems, in line with findings by Koren & Bell [5]. This application of NCF in the anime sector underscores the versatility and adaptability of AI in various domains, setting a precedent for future AI-driven recommendation systems.

Different from the Collaborative Filtering method which primarily makes predictions based on similar anime-watching patterns between users, the content-based method makes use of anime-specific information to make predictions. For example, if user A is fond of watching *horror* animes, then user A is more likely to receive *horror* animes recommendations rather than *ecchi* animes.

Many other recommendation algorithms have emerged over the years, demonstrating unique methodologies and promising capabilities. The Deep Knowledge-Aware Network (DKN) is one such model, which incorporates knowledge graphs to account for complex item relations, improving the accuracy of item recommendations [7]. It uses an entity embedding method to represent entities and relations, allowing for a more thorough understanding of items in relation to one another. FastAI's Embedding Dot Bias (FAST) model, on the other hand, introduces an additional bias term to the traditional matrix factorization approach, accounting for user and item biases that can significantly affect the rating process [6]. The bias term reflects any systematic tendencies for some users to give higher or lower ratings than the average, and some items to receive higher or lower ratings. This attention to

detail enhances the precision of recommendations. These are just a few of the many innovative recommendation algorithms shaping the future of personalized content delivery systems.

In this project, we first evaluate and compare the effectiveness of the existing collaborative filtering methods, namely *ALS* and *NCF*. Furthermore, propose a new content-based method *DTAR*, which particularly targets at the Kaggle Anime Dataset. Experiments are conducted to assess this method under different settings to test the robustness as well as accuracy. Last but not least, we apply Borda-Count voting technique in the end to combine the results from different methods.

The **novelty and contribution** of our project can be summarized as follows:

1) Perform meticulous data pre-processing techniques, such *Adaboost.SVR* on the dataset.
2) Propose a novel content-based method (*DTAR*), in which the user's specific preference and general trends are taken into consideration. We devise a special weight parameter to add them together.
3) We apply Bayesian rating in the last point as common rating.
4) *Adaboost.SVR* is applied to address the cold-start problem and attains relatively good performance.
5) By applying Borda-count voting method, we combine the result of the three algorithms (ALS, NCF, DTAR) into a hybrid answer.

II. Methods

A. Collaborative Filtering

Collaborative Filtering (CF) is a machine-learning technique commonly used in recommender systems. It is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many other users (collaborating).

The basic idea behind CF is that if two users have similar preferences or tastes in certain items, they are likely to have similar preferences in other items as well. Thus, the algorithm attempts to find patterns in how users interact with items and use these patterns to make recommendations.

In the following parts of this section, we will first introduce the matrix factorization for CF problems, and then demonstrate two CF algorithms: Alternating Least Square (ALS) and Neural Collaborative Filtering (NCF).

1) Matrix Factorization for Collaborative Filtering: Matrix factorization is widely used in recommendation tasks. Essentially, this technique seeks to identify hidden factors that represent intrinsic attributes of users and items in a lower dimension. That is,

$$\hat{r}_{u,i} = q_i^T p_u \tag{1}$$

where $\hat{r}_{u,i}$ is the predicted ratings for user u and item i, and q_i^T and p_u are latent factors for item and user, respectively. The CF algorithms are designed to find q_i^T and p_u that generate the ratings as close to the observed ratings as possible. To avoid overfitting, a regularization term is introduced. The regularized formula is given by the following,

$$\min \sum (r_{u,i} - q_i^T p_u)^2 + \lambda(||q_i||^2 + ||p_u||^2) \tag{2}$$

where λ is a the regularization parameter.

2) Alternating Least Square Algorithm: Due to the term $q_i^T p_u$ in the loss function, it is non-convex and applying gradient descent method can be computationally expensive. To address this issue, the Alternating Least Squares (ALS) algorithm was developed.

The fundamental concept of ALS is to optimize one of the latent factors q or p at a time while keeping the other factor constant. By doing so, the objective becomes convex and can be solved at each iteration efficiently. The alternating between q and p stops when there is convergence to the optimal.

3) Neural Collaborative Filtering Algorithm: Traditional collaborative filtering techniques such as matrix factorization often assume a linear relationship between user and item latent factors. However, in real-world scenarios, the relationship between users and items can be highly nonlinear. Traditional matrix factorization methods can struggle to capture complex patterns in the data due to their linear nature. To eliminate these limitations, the NCF algorithm fuses Generalized Matrix Factorization (GMF) and Multi-Layer Perceptron (MLP).

The GMF model utilizes an element-wise product to enable a more expressive representation, thus allowing NCF to capture more complex patterns in the data.

$$\hat{r}_{u,i} = a_{out}\left(h^T\left(q_i \odot p_u\right)\right) \tag{3}$$

Where the $\odot$ is the element-wise product of vectors. a_{out} and h stand for the activation function and edge weights of the output layer respectively.

Also, to obtain a high-level representation of the user-item interaction, the MLP component takes the concatenation of user and item latent factors as input and passes it through multiple hidden layers. For the input layer, there is the concatenation of user and item vectors:

$$z_1 = \phi_1\left(p_u, q_i\right) = \begin{bmatrix} p_u \\ q_i \end{bmatrix} \tag{4}$$

For the hidden layers and output layer of MLP, the details are:

$$\phi_l\left(z_l\right) = a_{out}\left(W_l^T z_l + b_l\right), (l = 2, 3, \ldots, L-1) \tag{5}$$

$$\hat{r}_{u,i} = \sigma\left(h^T \phi\left(z_{L-1}\right)\right) \tag{6}$$

In equation (5), W_l, b_l, and a_{out} respectively denote the weight matrix, bias vector, and activation function for the l-th layer's perceptron. When selecting activation functions for the MLP layers, there are various options to choose from, such as sigmoid, hyperbolic tangent (tanh), and Rectifier (ReLU). Given that the problem at hand is a binary classification task, the output layer's activation function is set to be a sigmoid

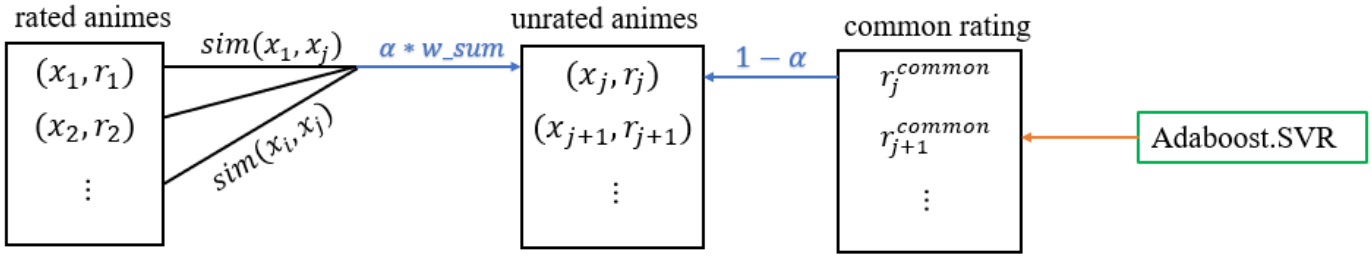

Fig. 1. An illustration of content-based method to predict the ratings of unwatched animes of an user. In the figure, x_i denotes the preprocessed feature vector for some anime i, and $sim(x_i, x_j)$ denotes the *cosine similarity* between feature vectors x_i and x_j. Our goal is to predict ratings for unrated animes, e.g., r_j, based on user-specific rating and the common rating r_j^{common}, obtained from Adaboost.SVR. For the unrated anime j in the above figure, $w_{sum} = \sum_{i \in \mathcal{A}} sim(x_i, x_j) r_i$, in which $\mathcal{A}$ is the set of rated animes for that user.

function, ensuring that the predicted scores fall within the range of (0,1).

NCF combines the GMF and MLP by concatenating the last hidden layer of them.

$$\hat{r}_{u,i} = \sigma\left(h^T \begin{bmatrix} \phi^{GMF} \\ \phi^{MLP} \end{bmatrix}\right) \tag{7}$$

Where ϕ^{GMF} and ϕ^{MLP} are obtained as follows,

$$\phi_{u,i}^{GMF} = p_u^{GMF} \odot q_i^{GMF} \tag{8}$$

$$\phi_{u,i}^{MLP} = a_{out}\left(W_L^T(\Omega)\right) + b_L. \tag{9}$$

where $\Omega = a_{out}\left(\ldots a_{out}\left(W_2^T \begin{bmatrix} p_u^{MLP} \\ q_i^{MLP} \end{bmatrix} + b_2\right)\ldots\right)$.

The likelihood function is defined as

$$P\left(\mathcal{R}, \mathcal{R}^- | \mathbf{P}, \mathbf{Q}, \Theta\right) = \prod_{(u,i)\in\mathcal{R}} \hat{r}_{u,i} \prod_{(u,j)\in\mathcal{R}^-} (1 - \hat{r}_{u,j}) \tag{10}$$

Where R represents the set of observed interactions, and R^- signifies the set of negative instances. P and Q refer to the latent factor matrices for users and items, respectively, while Θ represents the model parameters. By taking the negative logarithm of the likelihood, we derive the objective function to be minimized for the NCF method, commonly referred to as binary cross-entropy loss

$$L = -\sum_{(u,i)\in\mathcal{R}\cup\mathcal{R}^-} r_{u,i} \log \hat{r}_{u,i} + (1 - r_{u,i}) \log\left(1 - \hat{r}_{u,i}\right) \tag{11}$$

B. Content-Based Algorithm: DualTaste Anime Recommender

In this section, we present the details of our content-based recommendation algorithm (*DTAR*) for predicting user ratings for anime. The proposed algorithm predicts the rating by combining user-specific and common rating components. We use two separate models, $f(user, anime)$ and $g(anime)$, to compute these components. Below, we describe the algorithm and its components in detail, along with the associated mathematical formulations.

Fig. 2. An example of anime recommendation on Bilibili. In this screenshot, the latter two animes are rated, while the first two animes are unrated which is not desirable when making predictions.

1) User Rating: The user rating component, User Rating$(user, anime)$, is computed using a cosine similarity based method. Given an input anime, we first represent it as a feature vector. We then compute the cosine similarity between this vector and the feature vectors of all the animes that the user has previously rated. This results in a similarity matrix, which we multiply with the corresponding user ratings for these animes to obtain the user rating component.

Mathematically, let $\boldsymbol{v}_i$ denote the feature vector of anime i, and $r_{user,i}$ denote the user rating for anime i. The cosine similarity between the input anime a and each historical anime i is given by:

$$\text{Cosine Similarity}(a, i) = \frac{\boldsymbol{v}_a \cdot \boldsymbol{v}_i}{|\boldsymbol{v}_a||\boldsymbol{v}_i|} \tag{12}$$

The user rating component for the input anime a is then computed as follows:

$$\text{User Rating}(user, a) = \sum_{i=1}^{n} \text{Cosine Similarity}(a, i) \cdot r_{user,i} \tag{13}$$

where n is the total number of animes rated by the user.

2) Common Rating: Public rating, also known as common rating, plays a critical role in our model to predict the user rating given an un-watched anime. However, in many real-life

applications such as Bilibili, ratings are not always visible or provided as the Fig 2 shows. To tackle this challenge, we apply Support Vector Regressor (SVR), along with the classical boosting algorithm Adaptive Boosting (*Adaboost*), to generate the rating. SVR is derived from SVM, which solves a binary classification problem by formulating it as a convex optimization problem [2], by introducing an ϵ-insensitive region around the function. *Adaboost* is a special case of gradient boosting which uses exponential loss. In each iteration of *Adaboost*, we compute the next weak classifier by minimizing the weighted classification error, and α is learned adaptively. Repeat the same process until convergence [3]. The following algorithm shows a typical workflow of *Adaboost*.

Algorithm 1 *Adaboost* algorithm [3]

Input: $\mathcal{L}$, $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, T
  $H = 0$
  $w_i = 1/n$, i =1,2,...,n
  for $t = 1, 2..., T-1$ **do**
    $h_{t+1} = argmin_{h\in\mathbb{H}} \sum_{i:h(\boldsymbol{x}_i \neq y_i)} w_i$
    $\epsilon = \sum_{i:h(\boldsymbol{x}_i \neq y_i)} w_i$
    if $\epsilon < 1/2$ **then**
      $\alpha = \frac{1}{2} ln \frac{1-\epsilon}{\epsilon}$
      $H_{t+1} \leftarrow H_t + \alpha h_{t+1}$
      $w_i = \frac{1}{Z} exp(-y_i H_{t+1}(\boldsymbol{x}_i))$
    else
      Return H_t
    end if
  end for

Original Common Rating: By combining SVR with *Adaboost*, we are able to achieve a stronger ensembled learner with low variance.

Bayesian Rating: Due to various reasons, Original Common Rating may not achieve the most trustworthy and fair results. To try to deal with this challenge, we apply *Bayesian Rating* in our prediction process as well. The following expression describes the formula:

$$\hat{r}_i = \frac{NR + n_i r_i}{N + n_i} \tag{14}$$

where N is set to be the average number of viewers for each anime. The newly computed rating $\hat{r}_i$ will be treated as the new common rating.

To briefly sum up, the common rating component, Common Rating$(anime)$, is obtained using a pre-trained AdaBoost.SVR model, denoted as $g(anime)$. The input to this model is the feature vector of the anime, and the output is the common rating. We train the model using a dataset containing feature vectors of animes and their corresponding ratings.

Given a dataset $\mathcal{D} = (\boldsymbol{x}_i, y_i)$, where $\boldsymbol{x}_i$ represents the feature vector of anime i, and y_i is its corresponding rating, the AdaBoost.SVR model $g(\cdot)$ is trained to learn the mapping between the feature vectors and ratings. Once trained, the common rating for an input anime with feature vector $\boldsymbol{v}_a$ is given by:

$$\text{Common Rating}(a) = g(\boldsymbol{v}_a) \tag{15}$$

3) Final Prediction: We incorporate weights for both the user rating and common rating components to compute the final predicted rating for a user and an anime. Let ω_{user} and ω_{common} be the weights for the user rating and common rating components, respectively. In our algorithm, we first set $\omega_{user} = 0.5$ and $\omega_{common} = 0.5$, which means that user's own rating and the general trends contribute equally to the final anime prediction of this user. We also design a new way to calculate ω_{user}, given by the following formula:

$$\omega_{user} = \frac{N_i}{N_i + W} \times \eta \tag{16}$$

where W represents the average number of anime that is rated in general, and N represents the number of anime that is rated by user i. The hyperparameter η is used to control the contribution of the user's rating history. For example, if $\eta = 0.7$, it means that the maximum percentage of contribution for this user's ratings is 0.7

The final predicted rating is given by the following weighted sum:

$$\text{Predicted Rating}(user, anime) = \alpha + \beta \tag{17}$$

where $\alpha = \omega_{user} \cdot \text{User Rating}(user, anime)$ and $\beta = \omega_{common} \cdot \text{Common Rating}(anime)$. Also note that $\omega_{common} = 1 - \omega_{user}$.

By assigning equal weights to both components, our content-based recommendation algorithm aims to provide personalized predictions that strike a balance between user preferences and general trends in the anime ratings. The user rating component ensures that the prediction is tailored to the user's tastes, while the common rating component captures overall patterns in the data.

4) Cold-start Problem Solution: For a **new anime**, the traditional recommendation method cannot predict the result due to lack of rating history. Our proposed *DTAR* method can solve this issue by using *Adaboost.SVR* to compute the common rating of the new anime given its features. The predicted rating generated by *Adaboost.SVR* then can be used to calculate the user's intended rating on the anime even if the user has not watched it before.

For a **new user**, we use the off-the-shelf common ratings, either **original common rating** or **Bayesian rating**, to predict its behavior on the anime in our database.

C. Joint Recommendation

With the three recommendation algorithms (ALS II-A2, NCF II-A3, and DTAR II-B) at hand, we use a rating-to-score strategy to jointly use their predictions to get the final recommendation. The mechenisim is shown in Fig. 3. Generally, when a query regarding the $user_id$ enters, each algorithm first makes their top k recommends: $A_{ALS} = \{a_1, a_2 \cdots a_k\}, B_{NCF} = \{b_1, b_2 \cdots b_k\}, C_{DTAR} = \{c_1, c_2 \cdots c_k\}$. Then, for each animation in A, B, or C, we

accumulate its score based on its ranking in the lists: 1st position gets k points, 2nd position gets $k-1$ points, etc. At last, the joint recommender picks the top k scored items as the final recommendation. In our implementation, we pick $k = 10$.

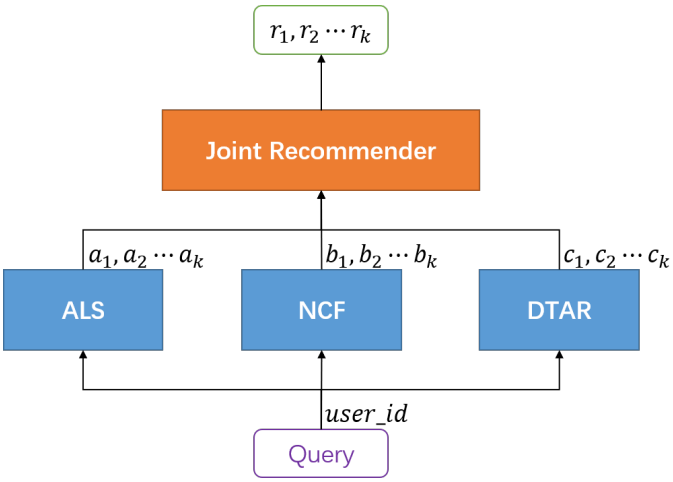

Fig. 3. Structure of the joint recommendation component. When a query enters, each of the three underlying recommendation algorithms gives k top choices to help the joint recommendation algorithm make the final decision.

III. Experiments

A. Dataset Description

We use the Kaggle Anime Recommendations Database (KARD) [link] to train, test and verify our recommendation algorithms. The KARD is a dataset containing information on anime shows and their ratings. The dataset includes over 12,000 anime titles, each with features such as title, genre, type (TV show, movie, etc.), number of episodes, and user ratings. The ratings were collected from the MyAnimeList website, where users can rate and review anime shows they have watched.

We chose the KARD as our data source since it has been widely used for machine learning and data analysis projects related to recommendation systems. Its size and diversity make it ideal for evaluating the accuracy and generalizability of our work comprehensively.

B. Data Preprocessing

For Collaborative Filtering methods, only the rating data is used, which is stored in *rating.csv*. The data file has three attributes, which are *user_id, anime_id*, and *rating*. Some users did not provide valid scores, which are marked as -1 in the *rating* column. To achieve good performance, we pre-filter out these invalid rows.

For Content-based methods, we need to preprocess *type*, *genre*, *episodes* attributes in *anime.csv*, and *rating* column in *rating.csv* to properly train the model and predict the result.

1) type: since this attribute is categorical, we need to convert each element in *type* into a numerical number. For instance, we change *TV* and *OVA* to 1 and 2 respectively. We apply the same method to other values in this column.

2) genre: each genre for an anime is a list containing one or more genre types. For example, anime *A* has *genre* value of "drama" and "romance". We scan through the whole anime list, make a new column for each attribute, apply one-hot encoding to each anime, and delete the original *genre* column. In the case of *A*, the newly created columns named *drama* and *romance* will be set to 1, while the others will be set to 0.

3) episodes: the number of episodes for some animes remains unknown, so we need to either get rid of them while training and predicting or estimate them using other values. Here, we adopt the latter approach. The unknown episodes value is estimated by the average value for that specific type. More precisely, let $\hat{r}_j^t$ denote a user j with unknown episodes value whose type is t, and let T denote a set of user indices with known episodes value whose type is t. Then,

$$\hat{r}_j^t = \frac{1}{|T|} \sum_{i \in T} r_i^t$$

4) rating: Different from collaborative filtering, we do not drop animes whose rating is -1. We estimate the rating of those animes by using equation (15). To be more precise, for a user i, the ratings for those watched but unrated animes are predicted by a similarity-based weighted sum from her rated animes and the public rating.

C. ALS Hyper-parameter Tuning

We do the hyper-parameter tuning on the regularization weight λ of ALS to find the best setting. The mean square error over both 2000 and 5000 users with different regularization weights are shown in Fig. 4. It is observed that $\lambda = 0.1$ achieves the lowest MSE.

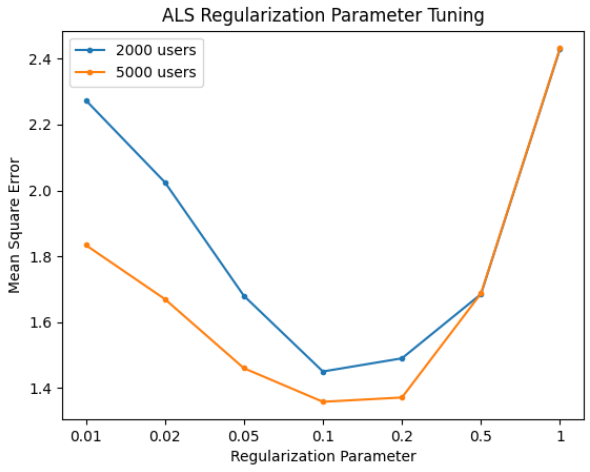

Fig. 4. Plot of ALS MSE vs. regularization parameter over 2000 and 5000 users.

D. Performance Comparison

We use the mean square error (MSE) to evaluate the accuracy of the three implemented methods. To test their performance under different data sizes, we run them on the

first 2000 users and first 5000 users subsets. The results are summarized in table 1.

TABLE I
MEAN SQUARE ERROR (MSE) FOR THREE MODELS

Method	First 2000 Users	First 5000 Users
ALS	**1.45**	**1.36**
NCF	1.92	1.89
DTAR	1.77	1.58

From the table, we find that *ALS* method achieves the lowest MSE both for the first 2000 users and first 5000 users, while the *content-based* method achieves the second best performance.

Figure 5 illustrates *DTAR* without applying Baysian rating under different parameter settings. Specifically, in *setting1*, $\omega_{user} = 0.5$; in *setting2*, the hyperparameter $\eta = 0.7$; in *setting3*, the hyperparameter $\eta = 0.5$; in *setting4*, the hyperparameter $\eta = 0.2$.

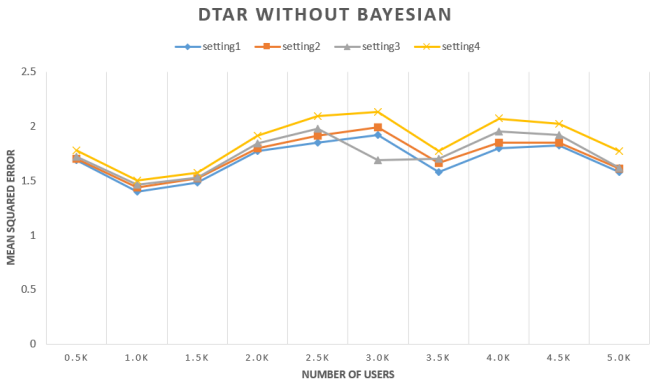

Fig. 5. The experiment result of *DTAR* without Bayesian under different settings.

Figure 6 illustrates *DTAR* with Bayesian rating. To be more specific, in *setting1*, $\omega_{user} = 0.5$; in *setting2*, the hyperparameter $\eta = 0.7$; in *setting3*, the hyperparameter $\eta = 0.5$.

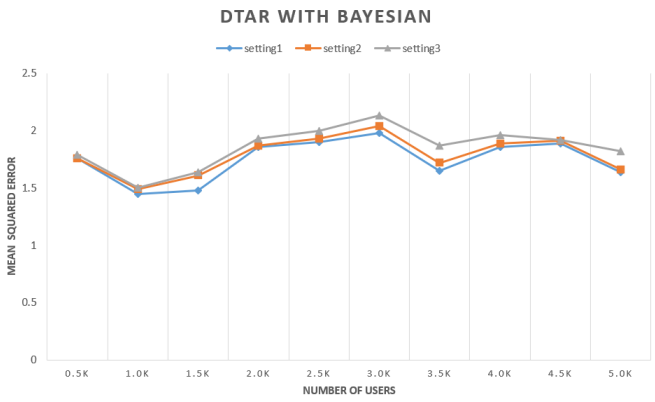

Fig. 6. The experiment result of *DTAR* with Bayesian under different settings.

From the figures, we find that without Bayesian rating, the minimum MSE is achieved when we set $\omega_{user} = 0.5$. Even though, the MSEs under the other three settings do not deviate too much from it. Another observation is that when we decrease η, the performance will also become worse, meaning that the user's own ratings are more crucial in predicting his/her own preferences. The similar situation also demonstrate in the experiment with Bayesian rating.

IV. CONCLUSION AND FUTURE WORK

In this paper, we introduced an animation recommendation system. Two types of popular recommendation algorithms, collaborative filtering methods and content-based methods, are investigated, implemented, and tested. We successfully adapted these algorithms on the Kaggle Anime Recommendations Database (KARD), which is a widely used dataset with the appropriate size and good diversity. Experiments show that our algorithms have good prediction accuracy.

We will continually put effort into this project to do more comprehensive experiments on the algorithms and try to improve their performances. Besides, we also plan to implement a user interface for piratical interactions and apply it to animation websites such as BiliBili, help to solve the real-world problems.

V. CONTRIBUTION

Details of contribution can be found below.

1) Dongzhuyuan Lu (24%): works on content-based algorithm.
2) Junhao Ran (24%): works on content-based algorithm.
3) Taimeng Fu (24%): works on collaborative filtering algorithm.
4) Yifan Wu (24%): works on collaborative filtering algorithm.
5) Yabin Cheng (4%): works on collaborative filtering algorithm.

REFERENCES

[1] Daxueconsulting, "The Colorful Marketing Potential Concealed in China's Anime, Comics, and Games (ACG) Market," Oct. 8, 2021. [Online]. Available: https://daxueconsulting.com/chinas-acg-market/. [Accessed: Apr. 22, 2023].
[2] V. N. Vapnik, "The Nature of Statistical Learning Theory," New York: Springer-Verlag, 1999.
[3] A. J. C. Sharkey, "Boosting using neural networks," Perspectives in Neural Computing, pp. 51-78, 1999.
[4] He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural collaborative filtering. In Proceedings of the 26th International Conference on World Wide Web (pp. 173-182). International World Wide Web Conferences Steering Committee.
[5] Koren, Y., & Bell, R. (2015). Advances in Collaborative Filtering. In F. Ricci, L. Rokach, & B. Shapira (Eds.), Recommender Systems Handbook (2nd ed., pp. 77-118). Springer.
[6] Howard, J., & Gugger, S. (2020). Fastai: A layered API for deep learning. Information, 11(2), 108.
[7] Wang, H., Wang, F., Zhao, M., Li, X., Zeng, D., & Chang, Y. (2018, April). DKN: Deep Knowledge-Aware Network for News Recommendation. In Proceedings of the 2018 World Wide Web Conference (pp. 1835-1844).