CSE 158 Assignment 2 Report: Anime Rating Prediction and Recommendation

Junhua (Michael) Ma
PID: A16299193
jum002@ucsd.edu

1 Introduction

Anime is a style of animation originated in Japan that is popular worldwide. Being fully animated, anime can be very creative and diverse in settings and content that make them appeal to wide range of audience. They can be lighthearted and comedic but also dark and gruesome, and any audience can find a set of anime that fit their interests most. Some genres and themes are explored more repeated and thoroughly in the anime world, and resulting in tens if not hundreds of similar anime such that enjoying one likely means enjoying all the other ones. Each person's tastes in anime can be very different which means popularity alone is likely not enough for a good recommendation, as some anime can be extremely popular but simply not appeal to everyone, like Dragon Ball Z or One Piece series. With the passage of time in the last few decades, the art form and style of anime was changing significantly, which result in people that prefer older or newer anime. Therefore, an effective recommender system stands out as a very useful tool for anime, and personalization or finding deeper associations may be more effective than very general ones.

2 Data

2.1 Dataset

The anime dataset comes from *myanimelist.net* API. *myanimelist.net* is a online platform for people to rank anime. After registering an account, users can select any anime and mark it as watched as well as providing a rating from 1 to 10. This is the most basic function of the platform and is captured by the dataset. The dataset has two parts. The first part is a list of all anime uniquely identified by an ID followed by other basic attributes of an anime including its name, genre, number of episodes, general rating, and popularity. The second part is a list of all ratings entries, each entry consist of userID, animeID, and rating. If the user has given a rating, it will be from 1 to 10. If the user simply marked the anime as watched but provided no rating, the rating will be -1. There are a total of 12294 anime and 7813737 ratings entries given by 73516 unique users in the anime dataset.

2.2 Data Preprocessing

The dataset is preprocessed to remove all abnormal entries with missing fields. All rating entries where rating is -1 is also removed for simplicity.

3 Problem Definition

The overall goal of this assignment is to create a rating predictor for unwatched anime for a user based on past ratings. The input to the predictor is a rating dictionary mapping anime to rating that represents the user's past ratings as well as a list of anime for the predictor to predict the ratings. The output of the predictor is a list of predicted ratings based on the input list of anime. The predictor will be trained on the training set, and use the learned representations to predict ratings for new users with a new set of ratings. To evaluate the predictor, MSE is used to evaluate the accuracy of the rating predictions. The reason for such a setup is largely motivated by the needs of immediate practical use, as the predictor can apply what it learnt from the dataset to predict how a new user would interact with each anime. This setup is also more general as the predicted ratings could be utilized to build a recommender, although the effectiveness of such a recommender is to be tested and is not the focus of this assignment.

4 Methods

4.1 Proposed Method

The method to build the rating predictor is to use similarity measures to predict ratings, where rating prediction for each user and item (anime) input is given by

$$r(u,i) = \bar{R}_i + \frac{\sum_{j \in I_u \setminus \{i\}} (R_{u,j} - \bar{R}_j) \cdot Sim(i,j)}{\sum_{j \in I_u \setminus \{i\}} Sim(i,j)}$$

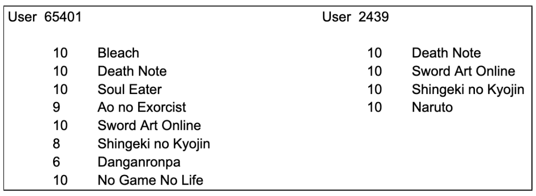
where $Sim(i,j)$ is a similarity function to compute the similarity between items i and j, I_u is the set of items rated by users u, and matrix R is a mapping from users to items. This straight-forward approach for rating prediction simply finds the users most similar to input user and average their ratings to the input item as prediction of the user's ratings. As a memory-based method, this approach has great potential due to the large amount of data available such that the predictor can reference when predicting ratings, and it's also simple and efficient enough overall to train and provide predictions. Most importantly, this approach readily fits the problem definition for the rating predictor since inputted user's rating history basically provides a new entry in I_u, as the predictor then incorporate the memory of users and items in training to predict ratings.

4.2 Similarity Computation

Computing similarity is an important aspect of the proposed method as the rating prediction is based on similarity. Jaccard similarity is used and it is computed as

$$Jaccard(i,j) = \frac{|U_i \cap U_j|}{|U_i \cup U_j|}$$

where U_i is the set of users that rated each item. In this case, the similarity between users is computed based on the items they rated, with higher similarity corresponding to more rated items in common. Although Jaccard Similarity tend to work better with binary data focused on user item interactions and other similarity functions like Pearson similarity would work better with quantitative data like rating, there are a number of advantages of Jaccard similarity function in this case. First, the Jaccard similarity function is simple and more efficient to compute as compared to Pearson similarity function. Efficiency is an important consideration as the anime dataset is massive. Second, Jaccard similarity works well with sparse data from the anime dataset, as there are a lot of anime and most users only rate a very small fraction of them, therefore many ratings are missing from each user. Finally, for each user, the number of ratings can vary considerably, which seems to lead to issues with the Pearson similarity function that may give high similarity to two users when they have a drastic difference in total number of ratings. On the other hand, Jaccard similarity focus on the intercept handles this issue better as well. Considering the nature of the dataset, the exact rating info that's ignored by Jaccard Similarity may also be of limited importance since two users having watched a lot of similar ratings should be pretty similar even if exact ratings differ. An example of most similar user pairing by Jaccard similarity is shown in **Figure 1**, in which the anime by the 2 users as well as their overall number of ratings are pretty similar, which is a reasonable indicator of their similarity in taste of anime and even usage of the *myanimelist.net* platform.



User 65401		User 2439	
10	Bleach	10	Death Note
10	Death Note	10	Sword Art Online
10	Soul Eater	10	Shingeki no Kyojin
9	Ao no Exorcist	10	Naruto
10	Sword Art Online		
8	Shingeki no Kyojin		
6	Danganronpa		
10	No Game No Life		

Figure 1: Example of most similar user pair using Jaccard Similarity function

4.3 Test Set and Evaluation

After the dataset is split into training set and test set where each entry is a tuple in the form of (userID, animeID, rating), the test set is constructed based on the setup defined in problem definition. For each unique user in test set, the ratings belong to the user is split in half, with the first half as the user history rating input, and the second half as items to predict the rating on. The labels are the user's ratings on items in the second half. MSE is computed and averaged for items of each user in test set, the final MSE computed is the average of MSE of all users. This testing setup directly test the rating predictor's ability to predict rating for new users based on a reference of their rating history.

4.4 Other Methods and Attempts

Other methods attempted include some model-based approaches like Latent Factor Model and Factorization Machines to predict ratings. Since these methods face the issue of cold-start, they don't directly fit into the predictor setup defined in the problem which inputs a rating history for a brand new user. Therefore, an idea is to use them solely as rating predictors for users in the training set, and use a similarity function to compute the most similar user to the input user, and simply use the model to predict rating of the most similar user as the final predicted rating for each input item for the input user. For this approach to work, both the similarity function and the model need to work well, especially with the model finding internal representations from user item interactions and even from features using factorization machines. However, my attempts to

Method	MSE
Baseline: Global Average	2.12
LFM (Bias only)	2.29
SVD (Surprise)	10.37
LightFM	14.25
Proposed: Similarity	**1.72**

Table 1: Result of each method in terms of rating prediction accuracy

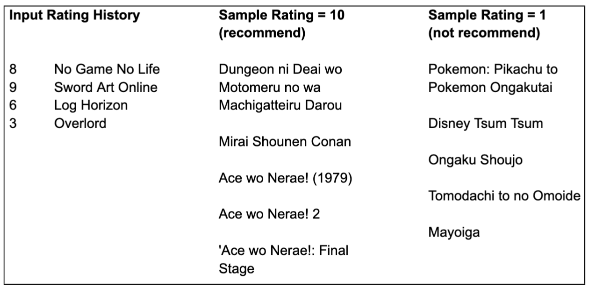
utilize model-based methods are generally unsuccessful as they are easily outperformed by the memory-based methods and even a simple baseline method to always predict the average rating. Initially, I tried to utilize factorization machines, but had difficulties running multiple factorization machine libraries including FastFM, Polylearn, and XLearn. The only factorization machine library I got working is LightFM, yet the resulting accuracy is outperformed with the model predicting 5 most of the time. Next, I looked to Latent Factor Models, using a simple Latent Factor model with bias only as well as the SVD algorithm provided by scikit-surprise library, yet the results are also not improved.

5 Results

The accuracy obtained from different models I attempted are shown in **Table 1**. The similarity rating predictor is the only method that I managed to get that out performs baseline accuracy, and I was unsuccessful with other methods. If the rating predictor is used as a recommender where given user rating history as input and run the prediction model on all anime not in user's rating history and output simply output the anime with the highest predicted rating, an example of the result is shown in **Figure 2**. As no features are used explicitly in the proposed method, it looks like incorporating features should offer some good improvements to the predictor as some features could be very effective for rating prediction like the time of release and number of episodes or seasons. Successfully utilizing model-based approaches like factorization machines could also improve the accuracy by finding more subtle associations in the data.

6 Related Work

The design of my work is initially largely referring to a type of music recommendation task known as automatic playlist continuation, which asks to build recommender systems that given an existing playlist, would recommend a number of songs to be added to the playlist. This helped form my idea of building a rating predictor that also looks at user's rating history as



Input Rating History		Sample Rating = 10 (recommend)	Sample Rating = 1 (not recommend)
8	No Game No Life	Dungeon ni Deai wo Motomeru no wa Machigatteiru Darou	Pokemon: Pikachu to Pokemon Ongakutai
9	Sword Art Online	Mirai Shounen Conan	Disney Tsum Tsum
6	Log Horizon	Ace wo Nerae! (1979)	Ongaku Shoujo
3	Overlord	Ace wo Nerae! 2	Tomodachi to no Omoide
		'Ace wo Nerae!: Final Stage	Mayoiga

Figure 2: Example output of a simple recommender system using similarity rating predictor

an input to predict ratings on other anime. Based on the analysis of top performing models in 2018 ACM Recommeder System Challenge for Automatic Music Playlist Continuation (Hamed Zamani, 2019), it appeared that the top performing models are mainly collaberative filtering matrix factorization models utilizing LightFM. In addition, the tasks are more in the form of playlist filling, as the test sets consists of incomplete playlist that the model should fill with recommended songs, and the model needs to learn as much representation as possible based on the given songs in each playlist. This insight opened up some possible modifications to my work, as I think that my setup, while easily applicable with the memory-based methods, may not be optimal for the application of model-based methods. With the simple item filling setup where useres information are ignored, this allows the construction of the matrix and the application of LightFM to be easier, whereas for my work it felt hard and non-intuitive to work with model-based methods. By simply analyzing and learning from just a list of anime, potentially with ratings, matrix factorization methods could work well in finding anime that fit well with the existing items, which can easily form a recommender system as well as be extended to rating prediction. Neural network is also used frequently to learn lower dimensional representation of items to improve the efficiency and effectiveness of the models, which leads to multi-staged systems starting with encoding stage to learn representation of items followed by matrix factorization to learn and make predictions. Going back to anime rating prediction, another existing study on anime rating predictor forms an insightful comparison to my work (Badal Soni, 2021). First, the anime dataset used is also originated from *myanimelist.net*, but with an additional users dataset with more information on users like gender, and their interactions with anime in more details including timesteamp data. As a result of this additional user data, a user-anime sparse matrix is constructed in the study to find connections between user and anime features. The task definition is also similar as it takes list of user ratings to predict user ratings on all anime in dataset. The method used in the study is a two-staged process to first learn lower dimensional embeddings through an autoencoder (collaberative filtering) and then use some form of clustering or nearest neighbor approach to to predict ratings (content-based filtering). This multi-staged approach combined with learning lower dimensional representation looks pretty similar to setups in the playlist continuation tasks, and again emphasizes the importance of collaberative filtering. The method for evaluation is mostly similar to my setup.

7 Conclusion

Overall, an interesting outcome is the effectiveness of the memory-based similarity rating predictor due to its simplistic nature, and it readily worked under my setup unlike the model-based methods. The setup allows the rating predictor to be highly practical in recommending anime for new users and it's a solid initial step, with the next step being to further improve the accuracy of the prediction through model-based matrix factorization methods.

The biggest disappointment is my failure to utilize model-based methods, and it could be caused by small programming mistakes on my part or a general setup that is not optimal for working with such methods. With the similarity-based rating predictor there are a lot of limitation, including the lack of features integration that makes factorization machines very effective.

References

Badal Soni, Nilutpal Nath Navarun Das Bhaskarananda Boro, Debangan Thakuria. 2021. Rikonet: A novel anime recommendation engine.

Hamed Zamani, Paul Lamere Ching-Wei Chen, Markus Schedl. 2019. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation.