COLLABORATIVE FILTERING: ANIME RECOMMENDATION SYSTEM

Liam Cannon, Yusra Suhail, Linda Tran

CSC 461: Machine Learning

Professor Christian Esteves

December 20, 2023

INTRODUCTION

Collaborative filtering is a method used to predict items based on similarity measures between users and/or items. It operates under the general assumption that users with similar interests tend to have common preferences for unseen items. One strong advantage of collaborative filtering is that the system will learn and identify patterns directly from the data provided by users' interactions. It can also help with recommending new items that the person is interested in. Another advantage is that it doesn’t require specific text, it only needs to know who interacted with that, basically who watched, bought, or rated each item. There are also disadvantages with collaborative filtering such as the issue that these datasets often suffer from data sparsity, leading to difficulty when it comes to suggesting new items or users based on historical data. For example, users may watch or rate only a handful of anime series and leave them with few or no ratings. Thus, when we try to compute similar users, they have no close neighbors and their recommendations are more prone to error. As is also a problem with K-Nearest-Neighbors, these models tend to be very computationally expensive. We will discuss some workarounds in our “Problem Definition” section.

To build collaborative filtering-based recommender systems, there are four techniques: memory-based, model-based, hybrid, and deep learning. For the purposes of our project, we will solely focus on memory-based filtering. The two primary types of memory-based collaborative filtering are user-based, and item-based, which are often used in tandem but have critical differences. User-based filtering makes recommendations based on other similar users who have had similar preferences on some set of items. In contrast, item-based filtering suggests items similar to other items based on how users have rated them. In our project, we only used explicit ratings, in which the users were asked for a rating and provided an answer, but there are also implicit ratings, calculated from purchases, clicks, views, etc.

The common distance metric to use for collaborative filtering is cosine similarity, which measures the cosine of the angle between two users' rating vectors in multidimensional space. The intuition behind this metric is that if they have rated the same set of items similarly, they should have a small angle, and should agree on new items. In practice, it is never this simple, as we will discuss later.

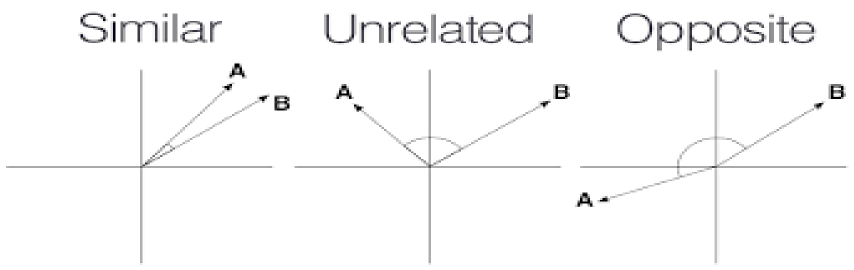

Figure 2.

Cosine Similarity Depiction

PROBLEM DEFINITION

Our goal was to develop a recommendation system tailored to an anime dataset. The primary goal is to build an advanced recommendation system for anime titles. We decided to work with both user-based and item-based approaches; when the user has sufficiently close users and has rated a certain number of anime, we will provide them a recommendation based upon what their similar users like (user-based), as this often gives more insightful recommendations than just item-based. This is because similar items are very often already related, such as episodes 1 and 2 in a series. However, when our users do not have any close neighbors, it is risky to provide a recommendation from them, so we resort to item-based filtering.

To remedy the issue of data sparsity we set a threshold for our selected users to increase the density of user ratings, but this has the cost of decreasing the number of users we train on. This was necessary as when we used Google Collab with smaller thresholds, it often crashed due to memory issues and long runtimes. Thus, we were restricted to larger thresholds in consideration of our time constraints. It is necessary to understand the data first however before manipulating it for our needs.

DATA

For our rating.csv dataset, we have the columns such as 'user_id', 'anime_id', and 'rating'. The 'rating' is a rating out of 10, which was explicitly given by the user. We started off with approximately 73,500 users, each user having an ID, and a list of anime they've seen, each anime with a given rating. If the user has seen the anime but did not assign a rating, the database stored this as a -1. This contains everything we need for the user-based recommendation, except for the actual name of the anime which can be found in anime.csv.

Initially, we expected to use the information in anime.csv, which contains attributes about the items (episodes, popularity, genre, etc…), but for item-based and user-based we ended up only pulling the name of the anime from this csv. This stands to prove that these techniques can work with very little information about the items themselves, but just about the user's interactions with those items.

PREPROCESSING

To calculate the similarity between users or items, we have opted to use cosine similarity. One problem with this approach is that there are often "tough" or "easy" graders, who rate all items harshly, or all positively. For example, if one user's mean rating is 5/10 (average), and the second user's mean rating is 7/10 (positive), an item that the first user rated a 5 should not be given to the second user as a 5 because that is a lower rating for the second user. The solution to this is mean centering; we subtract the mean of each user's ratings from each one of their ratings. Then, regardless of any user's mean rating, an average rating for one user will always translate to an average rating for another user, thus simplifying the calculation of the similarity between two users.

For normalizing the ratings, we begin by identifying the unique user identification in the dataset and their number of ratings. Any user who has given a -1 to an anime wasn't made a part of the total count of user ratings and it was also not a part of the mean. We changed the -1 ratings to the mean of that user assuming that if a person didn't rate an anime but watched it, they would have given it an average rating.

In order to compute cosine similarity between users, we needed a user interaction matrix, where the rows are the users and the columns are the anime. Aware of the fact the users didn't rate enough animes for the recommendation model to work properly, we decided on a threshold that was basically the minimum number of animes a user has to rate for it to be recommended animes by a user-based approach. The threshold reduced our data to 165 users from 73.5K users. Hence the threshold serves two purposes; it increases the quality of the dataset and it also decreases the computational cost. For the user interaction matrix, not every user would have watched every anime, so there were nans present in the matrix, we then imputed these NAN ratings with a 0 (as it will be the new mean).

METHODS

USER-BASED APPROACH

Once we have a user interaction matrix, we computed the cosine similarity between users that showed the user's similarity with other users. Using the users' similarity and user interaction matrix, we can recommend unwatched animes to a user and also predict their ratings. We can also use this matrix to predict ratings for animes that were watched by users just to calculate the error.

In order to give recommendations to the user, we used the user similarities to grab the similarity scores of this user with its 15 (decided by the elbow plot of number of nearest neighbors vs errors) nearest neighbors, where the nearest reflects the highest cosine distance current user has with its neighbors. Once we had this, we checked the top rated animes of these similar users and then checked which animes among these top rated animes were most common between all similar users. Top 5 animes were chosen based on their frequency of being watched among common users. We also made sure that the user wasn't recommended any anime that they had watched before. To do this, we created a function that finds the unwatched and watched anime for each user. We also had a function to predict the rating of a given anime and user, using the similar user's weighted ratings of that same item. The similarity score of each similar user was used to add a weight to the ratings of that similar user. Once we got this we did a sum of weighted average of these ratings and this will be our predicted rating.

To predict the error of our model, we tested our model against animes already watched by the user so we had something to compare to. We used root mean squared error for calculating error and then a mean of the individual errors for each anime watched by the user was calculated and reported. The total mean error of all users when the number of neighbors is 15 was reported as **1.323.**

ITEM-BASED APPROACH

The item-based approach means that we look at the interactions that the items received from the users. Here we look at how different anime titles are rated in the dataset. For this approach, the system calculates the cosine similarity for the anime titles.

The similar anime function is to recommend anime titles that are similar to the highest-rated anime by a user. It is set to return a list of five anime titles. The unwatched is the list of anime IDs that the user has not watched. The ‘watched’ is a list of anime that the user has already watched. Five recommendations are being returned and recommended to the user.

EXPERIMENTS

Our experiments were not as robust as we were hoping for, due to unexpected problems with Google Colab's resources, and the computational complexity of our functions. For instance, the threshold greatly influenced the runtime, and our runtimes were still very slow with only 165 users (using a threshold of at least 1000 anime rated). However, we were able to plot the number of nearest neighbors used and the resulting error below:

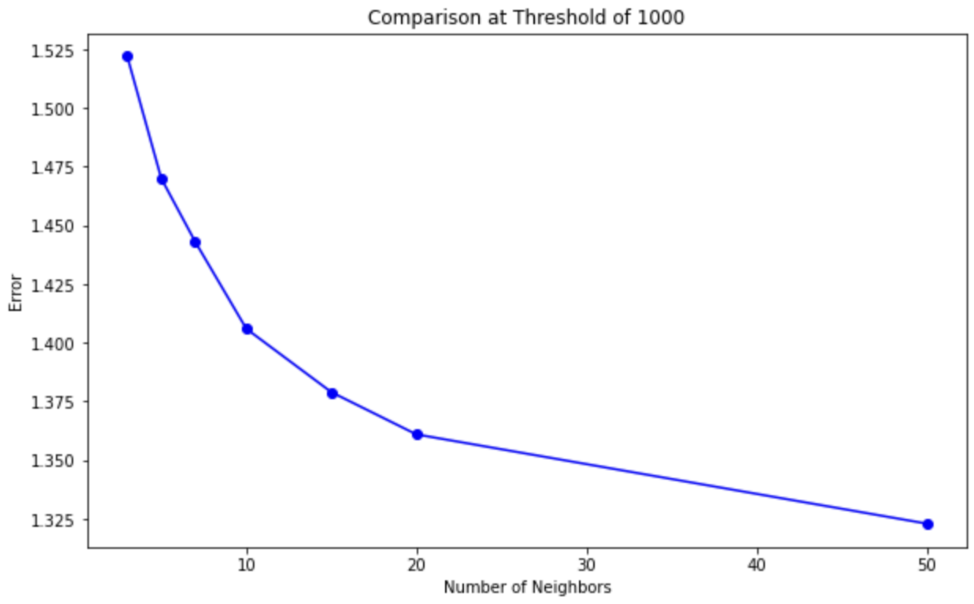

Figure 3.

Plotting error and number of nearest neighbors

CONCLUSION

In conclusion, we were able to use item-based and user-based filtering in tandem to create a recommendation system. Our RMSE was not as low as we were hoping for and likely would not suffice for a real world system, but there are steps we would like to take to improve it. First, we would like to do a more robust hyperparameter search , specifically the rating number threshold and the cutoff in which we switch to item-based filtering. We had to manually approximate the best cutoff, and the threshold was done mainly for time purposes, but we would like to do this exhaustively. Also, we want to look for better ways to handle the sparsity of the data, as imputing the mean for all missing ratings may not be ideal and potentially bias our results.

References

Ajmichelutti. (2017, March 3). *Collaborative filtering on Anime Data*. Kaggle. https://www.kaggle.com/code/ajmichelutti/collaborative-filtering-on-anime-data/notebook

Ayman, Zeynep Beyza. "Recommendation Systems: Collaborative Filtering." *Medium*, MLearning.ai, 15 Dec. 2022, medium.com/mlearning-ai/recommendation-systems-collaborative-filtering-75121f19dd03.

Benroshan. (2020, July 26). *Content & collaborative anime recommendation*. Kaggle. https://www.kaggle.com/code/benroshan/content-collaborative-anime-recommendation

CooperUnion. "Anime Recommendations Database." *Kaggle*, 21 Dec. 2016, www.kaggle.com/datasets/CooperUnion/anime-recommendations-database?select=rating.csv.

Dor. *Using Scikit-surprise to create a simple recipe collaborative filtering recommender system.* Data Science Portfolio. https://www.alldatascience.com/recommender-systems/simple-recipe-recommender-system-with-scikit-surprise/

GeeksforGeeks. (2023, March 31). *Collaborative filtering in machine learning*. GeeksforGeeks. https://www.geeksforgeeks.org/collaborative-filtering-ml/

Maklin, C. (2022, August 10). *Memory based collaborative filtering-user based*. Medium. https://medium.com/@corymaklin/memory-based-collaborative-filtering-user-based-42b2679c6fb5

Memetoglu, A. O. (2022, June 1). *4-) collaborative filtering and KNN*. Medium. https://medium.com/@aliozan_memetoglu/4-collaborative-filtering-and-knn-f997f8993256

IT Convergence. (2023, November 16). *Challenges & Solutions in building an AI recommendation system*. https://www.itconvergence.com/blog/challenges-and-solutions-for-building-effective-recommendation-systems/