Group Member(s):

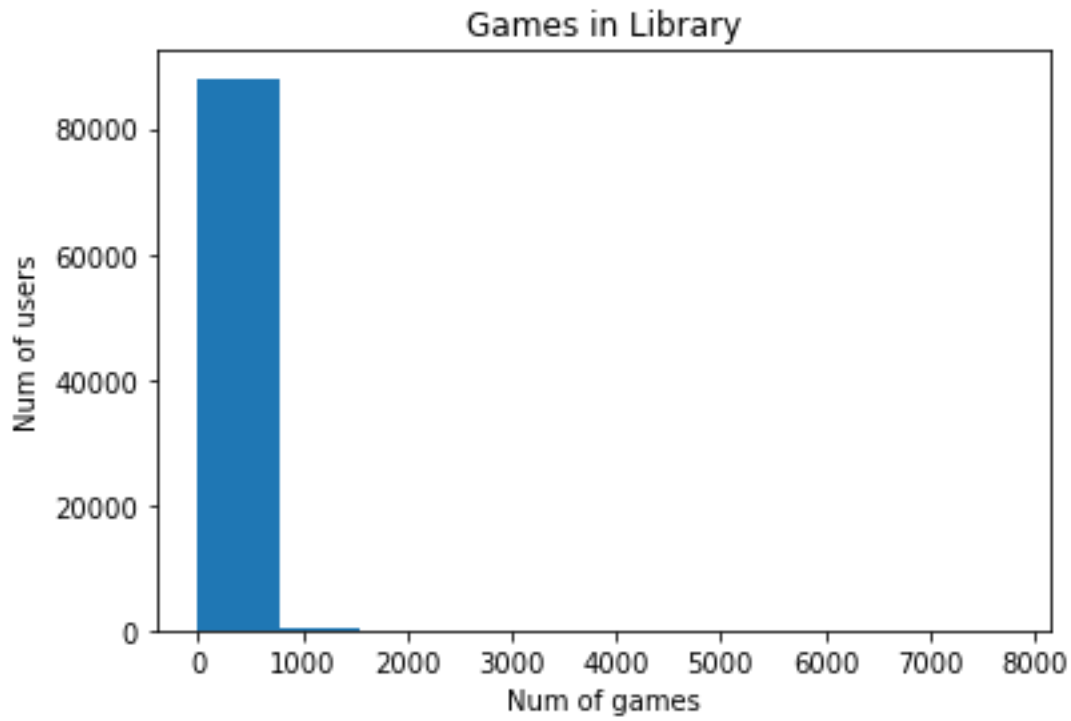
Dancheng Liu

Part I:

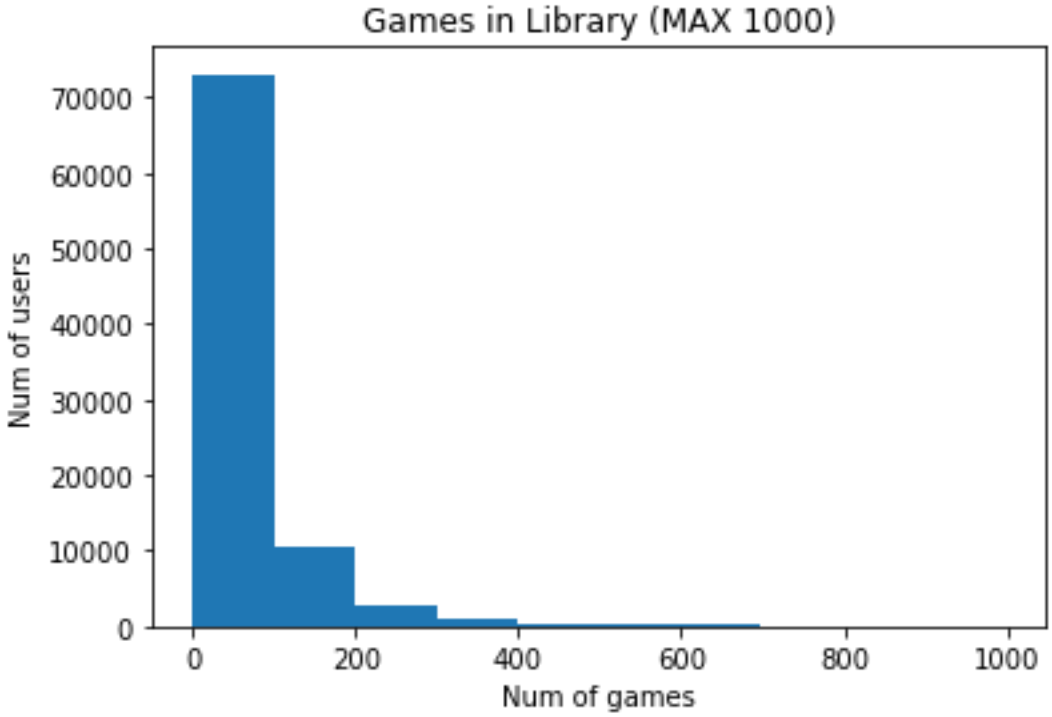
We decide to study the Steam dataset from
https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data

This dataset is a collection of five datasets on games distributed by Steam from the Australian server. We are interested in two of the five datasets: “Version 1: User and Item Data”, and “Version 2: Item metadata”. The first dataset includes games each user has, and the second dataset includes the description of games, including its genre.

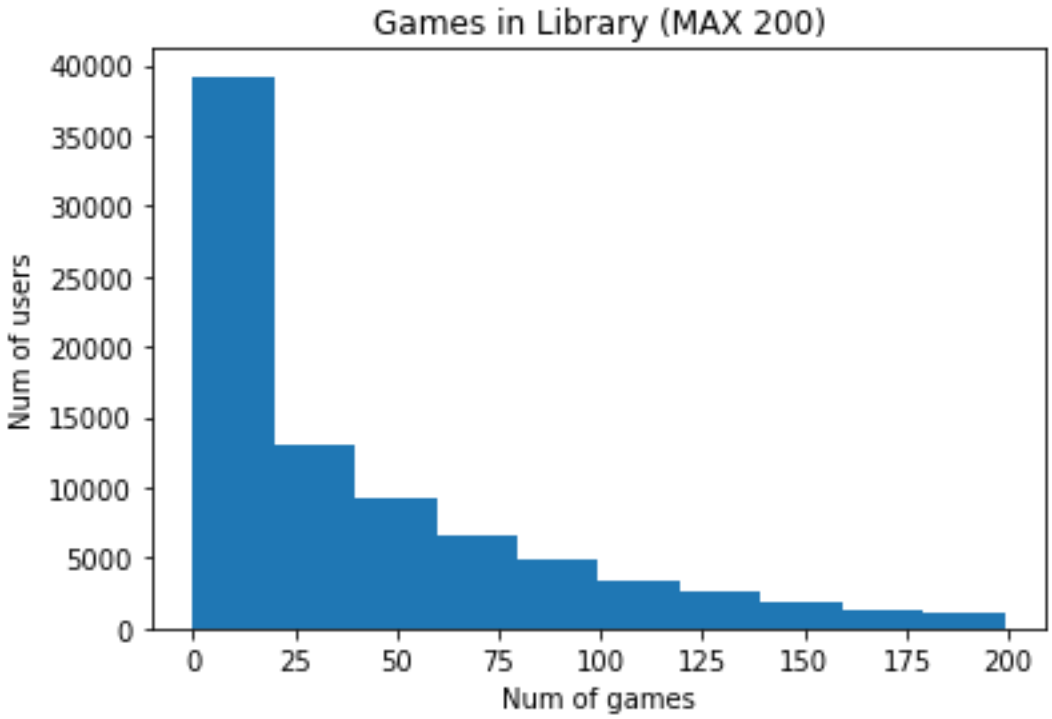
From the first dataset, we found that the number of games each user has is very different. There are 88310 users in the dataset, and among them, 16806 users have “no games” in their library (I will explain the quotation mark later”), while there is one user that has 7762 games in their library. The average games per user is 58.35. We tried to plot the relationship between user and number of games, and we have the following:



After excluding the outliers, we have:



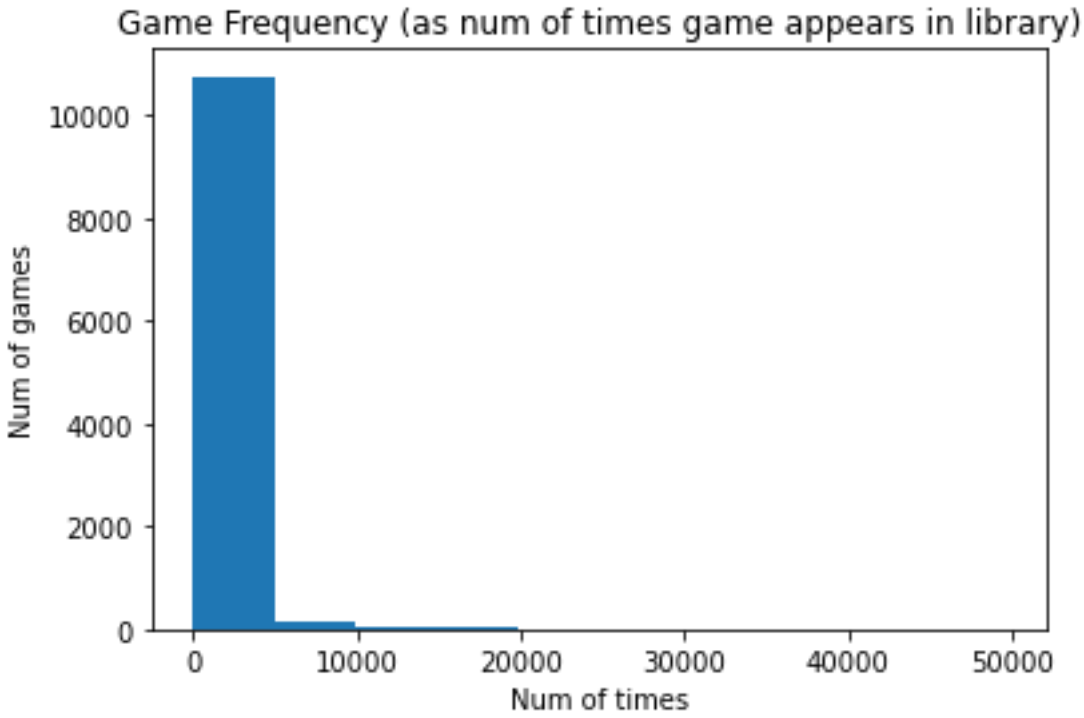
After further limiting the max number of games, we have:



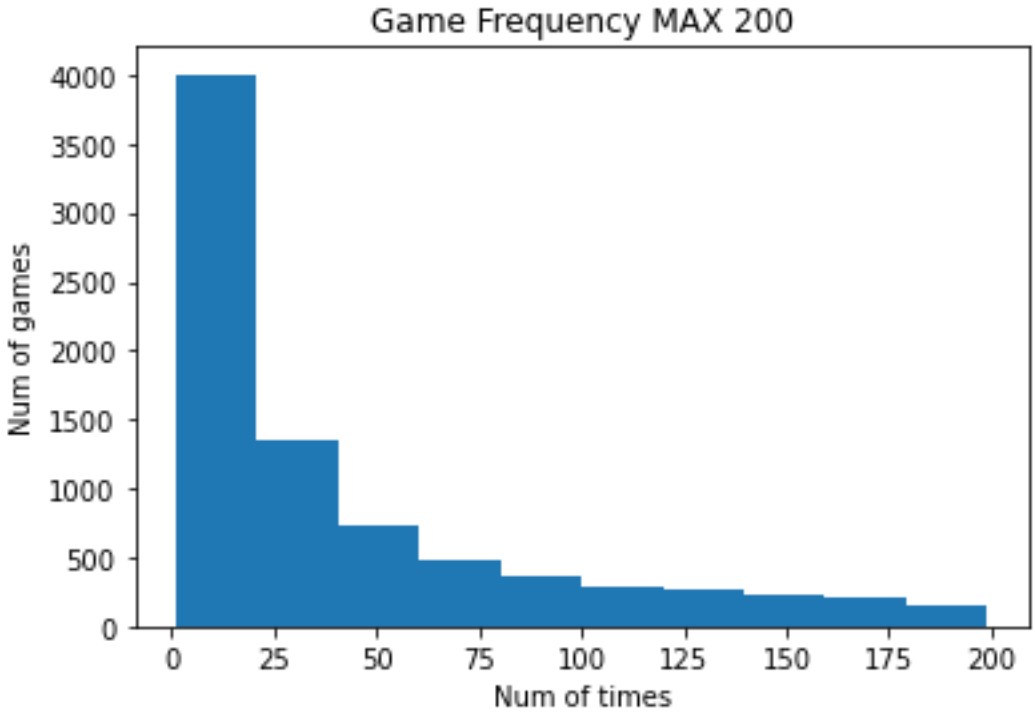
As we can see from the above graphs, the distribution is highly skewed. The majority of users have

Then, we performed analysis on the games on the second dataset, and we found that free games are not included in the dataset, which explains the cause of those users without any games in their library. There are 32135 games listed in this dataset, with the most popular one with ID “205790” occurring 49571 times in all users’ libraries. However, after further exploration, we found that this is actually an outlier. This game is “DOTA2 test”, which is a dependency of DOTA2. I have no idea why it appears in the dataset since DOTA2 is not included in the dataset (because it is free). The second most popular game is CS:GO, and third is Gary’s mode, which is all expected.

There are many games that not a single user buys, and there are 10978 games in total that are bought at least once. We plot the relationship between the number of games and the number of times each game is bought, and we have the following:



After limiting the maximum frequency to 200, we have:



As we can see, it is also highly skewed.

From the above analysis, we feel that we can create a recommender system to promote games to users based on their libraries, and this is what we do in the rest of this assignment.

Part II:

We can use a RS to predict whether a user would buy a game based on his existing library. Steam currently has a thing called discovery queue, which is basically a personalized promotion queue consisted of over 10 games. We could use knowledge we learned in class to construct a similar system. Adopted from the paper “Self-Attentive Sequential Recommendation” [1], we will use Hit@10, which measures whether the top 10 recommendations from our system hits one of the ground truths of the user’s library. The baseline to be compared against would be a popularity-based model, which would always promote the top 10 popular games to the user. Because of the high skew nature of the dataset, I expect this model to work very well. Excluding zero-game users and zero-buyer games, the entire dataset has 70912 users X 109768 games, and I decide to not do a simple train-test split because of the insufficient number of samples. Rather, based on the models for prediction, we will use different test method. All of the three models are trained on the entire dataset. Since our goal to predict a user’s preference based on his library, for the test dataset, I randomly hide half of the games in each user’s library and use the model to predict the hidden part based on existing half of the library. I will describe the reasons for choices over the three models in next part.

Part III and V combined:

Baseline

The baseline model is based on popularity. The accuracy is ~0.9557, which is a very high accuracy and is in line with the one from a 2018 paper "Item Recommendation on Monotonic Behavior Chains" [2]. Although the accuracy seems too high for a simple model, it is reasonable because of the high skewness in the data. Most users of Steam have some of the top10 popular games.

Model1: BPR

The first model I will use is BPR. It is suitable for this task because we are recommending games to users, and BPR is a very powerful model for RS. To verify the model, we will first train BPR on the entire dataset. Then, we will create the test dataset based on the given dataset, which contains all users but for each item the user has, instead of assigning 1 in the sparse matrix, we randomly assign 0 or 1 in the matrix. We recommend using BPR.recommend on each user and see whether the top 10 recommendations have a game that is in the hidden half.

I know this is not the most optimal way of verifying the model, and I do encounter some problems in the implementation. There are a lot of users who only a few games in their library. Take someone with only one game in library as an example, the test dataset will either put a 1 in the matrix for the only game he has, which means an automatic wrong prediction as BPR does not recommend what he already has; or a 0 in the matrix, which makes the prediction a pure cold start problem, which is not representative of the actual user's preference. Since from the explorative analysis, we know that the dataset is skewed, we can fairly ascertain that this is a big problem. However, even with this obstacle, the performance (Hit@10) on the raw model is still ~0.85, which is a very outstanding result. Some studies on the same dataset have better performance because of some preprocessing, and I will discuss it in Section IV.

After some tweaks on the hyperparameters, we were able to boost the accuracy on the test dataset to ~0.89. The hyperparameters are (factors = 3, regularization=0.1, iterations=200). It was discovered that this dataset needs a lot stronger regularization strength than the one in assignment 1, which means that the data is not as dependent on features. It is also noteworthy that since the way we construct test dataset is not optimal, increasing iterations might cause overfitting that we are not able to catch. This hypothesis can be only proven with a larger dataset.

Model2: GenrePop

The second model I will use is not so machine learning. I think that genres of the game would be a good prediction feature of what the user wants. Since each game has different genres recorded in the second dataset, we can predict the user's preferences based on the percentage of each genre in their library.

Similar to the first model, we use a half-given and half-hidden method for test. For each user, we train the model with all of their games in library, and for the test dataset, we randomly select a subset of their games to predict on the unselected subset. This test demonstrates the model's ability to predict new games based on existing games.

Using the two datasets, we first extract the genres of each game, and there are ~300 different genres. Then, for each user, we compute the percentage of each genre in their library. Based on that proportion, we used a greedy algorithm to recommend popular games of that genres to the user. The algorithm works as follows: recommend the most popular game under the most popular genre (measured by percentage) in the user's library. Then, reduce that genre's popularity and repeat the process until we have 10 recommendations. If the user does not have anything in the library (everything is hidden), we treat it as a cold start problem. The reason to use a greedy algorithm is because one game has multiple genres, and there does not exist an easy deterministic algorithm to maximize the recommendation accuracy.

This model serves as an alternative explanation of the dataset and an interpretation of the BPR model's latent factors. This model performs very well, reach ~0.95 accuracy. This result means that user is attracted to several genres they like. Although BPR model might catch this relationship, we do not know it for sure, and this model can tell us that information.

Model3: GenrePop-T

The last model is similar to model2 with additional temporal features. Everything is the same except that proportion of game genres now are associated with the amount of time the user has played the game. If the user has not played it before, it has a default weight of 1, and if the user has played it, we add a bonus weight with the formula (10*past_2_week+1*total). This model serves as an explanation that in addition to genres, users will tend to like the genres they are actively playing right now. However, it does not perform well for our task (reaching accuracy ~0.81). Although it is not high accurate, this does not mean it is a bad model. In fact, we could argue that it is actually a better model because of its attention on actively played games. This will not reflect on the test dataset we have in this report, but it is very important in real scenarios.

Some discussions on scalability and efficiency:

All three models are efficient to a certain extent. The main issue is the introduction of new features to the model, as it is not included in the training process. New features would mean new games to the first model, new genres for the second and third. Training is not as costly as I imagined on the first model, but it does take some time. The second and third model is simply calculations, and it takes constant time once we have the preprocessing done. Adding new features would not significantly cost any time, so they are much more efficient compared to a machine learning model.

Regarding scalability, I would say that Model3 generalizes the best over long-time intervals, and it is resilient to flavor changes in users' preferences. BPR might work, but it is too costly to constantly retrain the model every time something new happens.

Some further discussions on the models:

Due to time limit and device limit, I thought about some other models that rely on NLP, which need to use the review dataset. However, the small one does not have a lot of reviews in it, while the big one is over 1GB (too much for my poor laptop). So, I only made this prediction based on games' relationship with users, instead of a full prediction that involves game's reviews. I expect a model with reviews much stronger than the simple models I propose because of the culture in Steam reviews. As a Steam user, I notice that the binary "recommend" system is not representative of what the user actually wants to say. In particular, the rating system is heavily imbalanced with much more false "thumb-ups" compared to "thumb-downs". This makes analysis of the content of review critical.

I also thought about other models that are not suitable for this task. A classifier is not good for recommending, and KNN is not accurate enough for the task (plus it's super inefficient). I thought about using differences in proportions of genres between user's library and all games as a feature that represent the user's preference, but then I realized that it only works well for extreme cases. If a user does not focus on some small genres or one genre, the predictions are not accurate.

I also think that a machine learning approach for model2 and model3 would work better than the pure mathematical calculations, as it would potentially find the best weights to describe features that users use to determine their preferences. As suggested by my friend who studies in game industry, there are people who have a negative preference against games they recent play, which would lead to a negative weight on recent playing time in the model. (I do not know whether this is the general case, but machine learning approaches might figure it out automatically.) However, this is very costly in computation power (we are talking about at least 300+ features and indefinite number of LF). It would be useful in reality, but we need to be aware of the cost and scalability issue, as we need to retrain the model to remain accuracy, while a non-machine-learning model would only need some insertions and recalculations in the database.

Conclusion

This is a summary of performance of the four models (Baseline, BPR, GenrePop, GenrePop-T):

Model	Accuracy
Baseline	0.9557
BPR	0.86
BPR: Hyperparameter tuned	0.89
GenrePop	0.950
GenrePop-T	0.81

As we can see from this summary, baseline is actually the best. As I discussed above, this is because of the nature of the dataset, and it does not mean other models fail. A model with features being genres and popularity is also very strong; and adding a temporal feature to the model will make it resilient to flavor changes, which will make it strong against predictions in reality. If we preprocess the data, BPR model performs better than baseline (shown in Part IV), which indicates that BPR model is more accurate for users after cold-start.

Part IV:

Based on how my discovery queue (result from their RS) is presented, the current strategy Steam uses might be very similar to the third model + a similarity model between games (using some sort of similarity measurement of games based on users that buy them). I think this is a reasonable choice because of efficiency, and based on the implementation, it works good enough.

There are many literatures studying RS for Steam, while [1, 2] are highly relevant to this dataset. My results align closely with "Item Recommendation on Monotonic Behavior Chains" [2]. We have the same baseline accuracy, and in their paper, they say that BPR model after preprocessing has accuracy of 0.963. The preprocessing stage removes all users with less than five games in their libraries. This effectively avoids the problem I described in the BPR model section. Their proposed monotonic behavior chain model has accuracy of 0.968, a slightly higher result than BPR. In "Self-attentive sequential recommendation" [1], the implementation might be different as their popularity-based RS has accuracy of 0.7172, and their proposed model, self-attentive sequential recommendation, has accuracy 0.8729.

In recent years, there are deep learning approaches to the Steam's prediction datasets (although I do not know whether they are using the same dataset). In "Using Deep Learning and Steam User Data for Better Video Game Recommendations" [3], they have a nice table summarizing the results. [4] reaches 0.876 MAP, while their paper reached 0.895 MAP.

Clearly, DeepNN approaches do not improve the state-of-art accuracy on the recommender system. This is normal as from this report, we see that simple models already gives almost perfect predictions. This means that the features of user's preferences are very simple, and DeepNN would only worsen the predictions by over-complicating it.

References:

1. **Self-attentive sequential recommendation**
   Wang-Cheng Kang, Julian McAuley
   ICDM, 2018
2. **Item recommendation on monotonic behavior chains**
   Mengting Wan, Julian McAuley
   RecSys, 2018
3. Dylan Wang, Melody Moh, and Teng-Sheng Moh. 2020. **Using Deep Learning and Steam User Data for Better Video Game Recommendations.** In 2020 ACM Southeast Conference (ACMSE 2020), April 2–4, 2020, Tampa, FL, USA. ACM, New York, NY, USA, 6 pages. h ps://doi.org/10.1145/ 3374135.3385283
4. G. Cheuque, J. Guzmán, and D. Parra, **Recommender Systems for Online Video Game Platforms: The Case of STEAM**, in Companion Proc. of the 2019