Recommender System Based on Video Games Data: A Survey

Explore the trade-offs of recommender systems based on different models

Shantao Yi
UC San Diego
9500 Gilman Dr.
La Jolla, California 92093
shy146@ucsd.edu

Cheng Gong
UC San Diego
9500 Gilman Dr.
La Jolla, California 92093
c5gong@ucsd.edu

Junchi Zhou
UC San Diego
9500 Gilman Dr.
La Jolla, California 92093
juz083@ucsd.edu

ABSTRACT

Nowadays, digital entertainment plays an really essential part of everybody's life, especially for games like computer or console games. Therefore, giving recommendations to people and helping them explore new games will be a very profitable and growing industry within this field. In this paper, we will explore different approaches for a video game recommender system as well as looking into the state-of-the-art methods currently being used.

KEYWORDS

Recommender system, purchase prediction, video games, Steam

ACM Reference format:
Shantao Yi, Cheng Gong, and Junchi Zhou. 2018. Recommender System Based on Video Games Data: A Survey. In *Proceedings of CSE 158 Assignment 2, La Jolla, California USA, November 2018 (CSE158'18),* 6 pages.
DOI: 10.475/123_4

1 INTRODUCTION AND EXPLORATORY ANALYSIS

Introduce the dataset we will be using as well as perform an exploratory analysis on the dataset.

1.1 The Dataset

We decide to use the **Steam Video Game and Bundle Data** from Prof. McAuley's website (https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data; official citations done below). This dataset contains three different sub-datasets:

(1) a (user, items) dataset where each entry is all the items (games) ever bought by a user, as well as the time they spent on each game
(2) a (user, reviews) dataset where each entry is all the reviews done by a user.
(3) a (bundles) dataset where each entry is all the information of a Steam game bundle.

We will be only using sub-datasets (1)[1] and (2)[2] to complete our task (specified below) and they contain over 50,000 entries each thus qualifies the requirements.

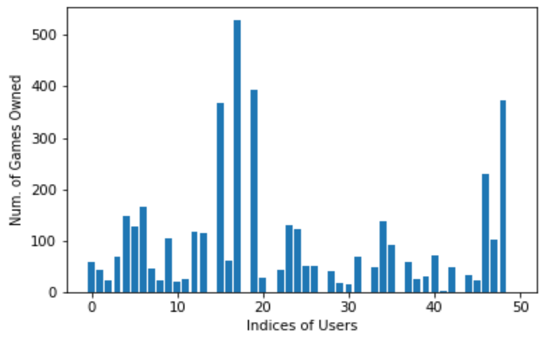

Figure 1: Num. of games owned per user

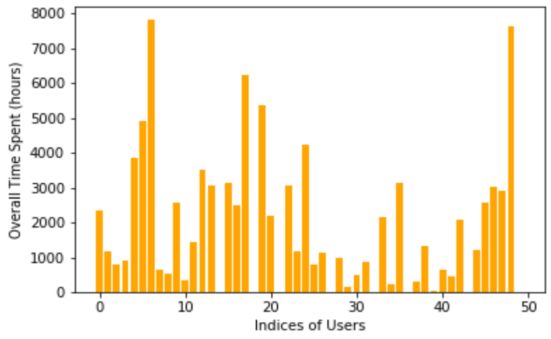

Figure 2: Overall time spent on games per user

1.2 Exploratory Analysis

1.2.1 Sub-dataset I. Firstly, we may observe rather easily that this dataset contains **88310** entries, **87626** users and **10978** items (games). We also find out that the average number of games owned by a single user is roughly **58**, which is a quite astonishing.

Nevertheless, the average may not be a very good indication since the number of games owned by each user[1] actually varies quite a lot (as shown in **Figure 1**) from as large as over **500** games

[1]the two above graphs show the first 50 users in data sorted in alphabetical order of their names so that the graph stays clean and the trend can be easily observed.

© 2018 ACM. 123-4567-24-567/08/06…$15.00
DOI: 10.475/123_4

Table 1: Popularity of games based on user ownership

Game Name	Num. of Owned Users
Dota 2 Test	49571
Counter-Strike: Global Offensive	43776
Garry's Mod	43301
Unturned	38682
Left 4 Dead 2 Beta	37044
Left 4 Dead 2	37044
Terraria	29239
Warframe	25807
Portal 2	24465
Counter-Strike: Source	24220

to **0**, owns nothing. Yet we can still observe that most of the users owns **less than 180** games.

Furthermore, we may observe (from **Figure 2**) that the overall time users spend on playing games varies a lot, too. By comparing **Figure 1 & 2** we may also find out that the number of games owned by a certain user mostly forms a **(positive) linear relationship** with the total time they spend on playing them, thus may be a relatively good indication in most cases. However, this is not always the case, that is, a user may own a lot of games yet plays them rather rarely (like the 16_{th} user), and may own only a few games yet plays them intensively. (like the 7_{th} user). But still, overall for most users the (positive) linear relationship still exist.

Now we start to look at aspects on the game side. First of all we counted for each game, how many user actually owns it, and sorted out the **10** most popular games within the dataset (as shown in Table 1). Yet from this table we can see that some "games" are actually test or beta version of the actual game (for example "**Dota 2 Test**" and "**Left 4 Dead 2 Beta**"). These entries may add noise to our dataset since most of them are bundled together so the user who owns the actual game will automatically own those versions, and will never touch them because they are mostly for testing or trouble-shooting. Therefore, when we are performing our prediction task we may want to take extra care of these entries.

However, simply counting the number of ownership may not tell the whole story. **Figure 3** shows the total time all the users spend on certain games and we can clearly see that **Counter-Strike: Global Offensive** and **Garry's Mod** dominated half of the pie chart, whereas all other popular games (**18** of them!) only consumes half of the time. Therefore, when predicting we should treat time spent on games very seriously since they provides the information of whether this game is actually popular or not.

1.2.2 Sub-dataset II. Firstly, we observe must-have statistics, that is this dataset has in total **25799** users and in total they conducted **59305** reviews. Therefore, on average each user conducted roughly **2** reviews.

For each review, there is a Boolean flag representing whether the user recommend this item or not. However, we observe that **88.48%** of the reviews are positive, that is the user is recommending the game. We may come to the conclusion from this that most of the users tend to only give a review when they actually liked the game, and only a few actually review a game they does not like. This

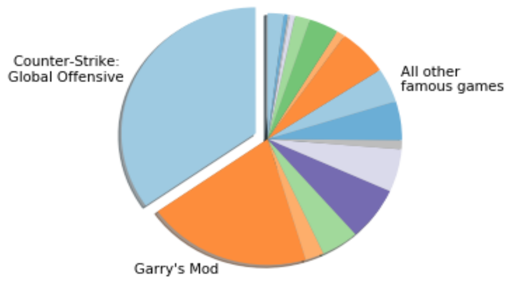

Figure 3: Time spent from all users on popular games

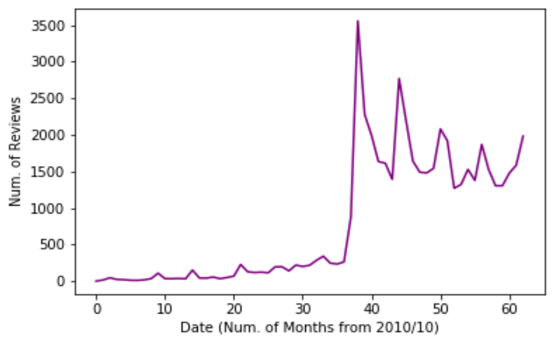

Figure 4: Num. of Reviews Conducted as Time Involves

might be an important factor to consider if we want to integrate the Steam recommendation system into our own predictor.

Furthermore, we look at the influence of time stamps regarding the number of reviews. By creating and observing **Figure 4**, we surely can notice the outburst of reviews during around 40 monthes after Oct. 2010, which is around **2014/2**. This is indeed roughly the time when computer games are becoming more and more popular, so does Steam. After that, there is a reviews outburst roughly every year, which also make sense since new games may be released every during this time. Therefore, if a game is released before year 2014, the reviews may not provide a great prediction due to the lack of amount, and vise versa for games that are released after year 2014.

2 PREDICTIVE TASKS

The predictive task we will be preforming is: *Predicting whether a certain user will purchase a certain game or not, based on their past purchases and playtime of the games, past game reviews given by the user, and if other similar users have purchased this game or not.*

2.1 Model Evaluation

Firstly, we will split all of the available data into three one third's, acting as training data, validation data, and testing data. For validation and training data, we will evaluate the performance of the model by predicting on the (user, game) pair, whether the user is going to purchase the game or not; half of those data will be that user has purchased the game (so that the model wants to predict 1), and the other half is randomly generated pair where user has not purchased the game (so that the model wants to predict 0).

To fine tune the model using validation data, we try to fit the model with the correct parameters so that the accuracy of the model can be high while not overfitting the given training data. For testing and evaluation, if much more than 50% of the predictions match the actual purchase history of the (user, game) pairs in the test data, then our model is significant.

Also, a great baseline model that can be used for evaluation will be to simply return True if the game is among one of the most famous games, and false otherwise. In games players community, most people usually pick the most played games by the whole community. The general popularity plays a huge role in affecting users' decisions to buy or not to buy the game. Therefore, if the game is in the first half of most popular games, we predict all users will buy it (more details in Section 3)

2.2 Features that might be relevant

To predict a purchase, we will extract several features about this user regarding all other users and games. The features that might be relevant to the task at hand will be the following:

- **Simple: Number of Purchases**
  If the user has purchased a large number of games, they will have a high likelihood of purchasing another game, since they are more comfortable with the idea that purchasing a game on Steam will bring positive effect to them in some way. Therefore, they will not have a hard time convincing themselves to purchase more games, due to all the games they have previously purchased.
  To retrieve this feature, we can simply use the **sub-dataset I**, find the specific user, and note the item count property of that user, or count the length of the user's item list.
- **Simple: Total playtime**
  If the user has spent a lot of time on games in Steam, then they will likely to invest more on purchasing another game, since they trust this platform and the quality of the games on Steam. Although this may not be exactly true since we have observed from our analysis above that a few users do stick with a few games yet still commits huge amount of time on it.
  To retrieve this feature, we can simply use the **sub-dataset I**, iterate through the item list of this specific user, and sum up all the play times each user spend for each of their games.
- **Simple: Playtime per Game**
  If a certain game has been played for many hours (comparing to all the other games on Steam), the game will very likely be purchased by users based on the popularity of this game.
  To retrieve this feature, we can simply iterate through the **sub-dataset I** and sum up all the play times of all users on this certain game. To compare to all the other games on Steam, Find the ratio of the play time on this certain game and the play time over all users and all games.
- **Simple: Number of Purchase**
  If a certain game has been purchased many times by many different users relatively more than other games do, the game will very likely be purchased by the other users, because in general this game is very popular and will be likely to be purchased.
  To retrieve this feature, we can simply iterate through the **sub-dataset I**, and count the number of users who owns this certain game.
- **Simple: Number of Recommendations**
  If a certain game has been recommended many times by many different users relatively more than other games do, the game will likely be purchased by other users because this game is highly recommended and people do believe in other people's recommendations.
  To retrieve this feature, we can simply use the **sub-dataset II**, and count the number of reviews on this certain item that has a positive recommendation.
- **Complex: Similarity Between This Game and Other Games Purchased by This User**
  If a certain user has purchased similar games before, they will very likely to have a tendency of purchasing this game because this type of games looks similar and will attract the users to purchase them the same way.
  To retrieve this feature, we can use both the **sub-dataset I** and **sub-dataset II** to calculate the Jaccard Similarity between items; or we can use review texts and implement TF-IDF for the similarity.
- **Complex: Past Reviews on Whether The User Recommend Similar Games**
  If the user really like and recommend a certain type of games, they will very likely be drawn towards purchasing another similar game.
  To retrieve this feature, we can use both the **sub-dataset I** and **sub-dataset II** to calculate the Jaccard Similarity between the current item and all items this user reviewed.
- **Complex: Playtime of Similar Games**
  If the user really like and have played a lot of similar type of games that they purchased before, they will likely be drawn towards purchasing another similar game.
  To retrieve this feature, we can use both the **sub-dataset I** and **sub-dataset II** to calculate the Jaccard Similarity between the current item and all items this user have played intensively (based on a threshold of playtime).
- **Complex: Purchase History of Other Similar Users**
  If lots of users similar to this user have purchased this game before, then this user will also be likely to purchase this game; if most of them have not, then this user will be unlikely to purchase this game.
  To retrieve this feature, we can use both the **sub-dataset I** and **sub-dataset II** to calculate the Jaccard Similarity, or see if they recommend the game similarly using Cosine Similarity, or have played a lot of this similar type of games before using Pearson correlation. A user who have the

same tendency of purchase, recommend, and playtime will be considered as a "similar user".

- **Complex: The Most-recently Played Games**
  People tend to follow trends. Therefore, if a game is being played by lots of users in very recent time, it is very likely that this game will also be purchased by this user.
  To retrieve this feature, we can use both the **sub-dataset I** and **sub-dataset II** and, in each of purchased items, there are both hours which are played time forever and played time in the last two weeks. Using such data to calculate the recently most played items, we can suppose that a large group of users purchased or would purchase those items.
- **Complex: Game Names**
  Since we are not given the categories of the games, we can predict the similarity of the games based on the names of games. On one hand, related games might use similar descriptive words. On the other hand, different versions of games might share the same game just with minor difference. By finding the similar games through names, we can recommend the user with the ones that are near to their purchased items.
  To retrieve this feature, we can use both the **sub-dataset I** and collect all the game names; then we seek similarity between them by checking if they contains the same title and etc.
- **Complex: Single-player or Multiple-player**
  According to most player's habit, they always mention their team players in the review if it is a team based game or solo game. We suppose users will prefer similar kind of games based on the number of players. Even though we are not given the information directly, we can extract it from the review text to hypothesis if the user likes sole or team mode and if the item is sole or team mode.
  To retrieve this feature, we can use both the **sub-dataset II** and extract all review texts. Within these texts, we give a extra weight to words like "teammates", "competitive" so we can differentiate these two type of games.

3 MODEL SELECTION

After coming up with features that might by useful, we realized that the general directions we can explore, based on those features as well as the property of this dataset, will be: Similarity-based models and Text-based models. Therefore, we got our hands dirty, tried out few different models from these fields, and eventually chose the model with the best performance.

3.1 Our Model

The model we decided to use as our final model is a model which utilizes a modified version of Jaccard Similarity, based on which users have purchased the items as follows. For each item, if its set of users who purchased it highly overlaps with another item's set of users who purchased it, then these two items are similar. This overlapping part of two items decides them to be similar using a tuned similarity threshold. Then, if enough items similar to this item at hand are purchased, then we will predict that this use will purchase this item as well. This composes as our second tuned counting threshold. The accuracy of the test data after tuning is around 84.93%.

3.2 Optimization

Our model follows the idea to predict that the user will purchase the item if this user has purchased enough similar items. However, in order to define what items are similar to this item at hand, we need a Jaccard Similarity threshold between 0 and 1 to determine whether another item is similar to this item. We first took an educated guess of 0.15 based on a previously purchase prediction task we have seen with a set of Amazon clothes purchases. Setting up a loop to iterate through the possible similarity threshold towards a threshold where the prediction accuracy on validation data is the higher than the current produced accuracy. Once reached a stable accuracy, where it does not change much even if you change the similarity threshold, we move on to determine the next threshold, the count threshold.

Note that after we find a list of similar items, we need a count threshold that will means that enough similar items have been purchased by this user, so that the user will purchase this item. The guess on this threshold is initially 0.2, which yields us a 50% accuracy. This means that the threshold is either too high or too low. After step-by-step tuning the count threshold, we find the count threshold yields the best validation accuracy at 0.0015. After continuous tuning, the best prediction accuracy on validation data has reached 0.8510515966. The accuracy of the test data is around 0.8492646213.

3.3 Other Models

3.3.1 Baseline Model. Our baseline model is to predict whether the user will buy the game or not based on the popularity of the game. We ranked the popularities of games using the number of users who have purchased it. In the prediction phase, if the item is in the top 50% of the most popular games of the training data, we predict that the user have purchased this game. The baseline model prediction accuracy on our prediction set is around 67.97%.

3.3.2 Cosine Similarity-based Model. This model is based on calculating user similarities using modified Cosine Similarity which is using each of the users' purchased games and their total playtime on each game since they purchased the game. The idea is that if both users have spent a longer time playing the same games and the sets of purchased items overlaps more, they are said to be more similar. Specifically, we compute vectors of the total playtime of each game that each user plays, and compare two users and calculate their similarity by computing the cosine similarity of the two vectors of those users. We intend to find the list of most similar users, and if a high ratio of them have purchased this specific item, we will predict that this specific user will purchase this item as well. The accuracy of the test data after tuning is around 79%.

After step-by-step tuning each of the threshold parameters with validation data, the accuracy on validation data has reached 81.41%. The accuracy of the test data is around 79%.

3.3.3 Text-based Model: TF-IDF with SVM. This model is based on trying to train a SVM to create the hyper plane dividing purchases and non-purchases. First of all, for each user we extract all of their past review texts and combine them all together. With all these text data extracted, we use a TF-IDF classifier to create our X for the SVM. As for the y, we simply use a list of 0's and 1's representing non-purchases and purchases. With both X and y prepared, we throw them all into our SVM to train, and then predict with validation and test datasets (creating X's using the same ways above).

We also tried adding game names as well as adding in other features like recommend or not with a weight of whether other people find it helpful or not. Yet non of our attempts gained us a reasonably good result, with the best one still not so far aways from our baseline model.

3.4 Strengths & Weaknesses

Our model made predictions considering personal preferences by recommending items purchased by other users who are similar to this user based on all game players' buying histories in our data set. Every user will get their own unique recommendations instead of the universal ranking of all games. Other works such as Amazon proved that using a personal recommender system can improve the company sales by 35%. In our way, the unpopular content can also be recommended, relying on a personalized recommendation system based on user habits data - using personalized recommendations, compared to the simple display of the most popular list of games.

The first weakness of our model lies in the cold start problem. For users who are new or not played Video games often, our recommendation fails to find similar items for them based on their personal profiles which are not included in the data set. In order to address the problem, we want to have user's features such as gender and age to calculate another dimension of similarities among users. The second drawback in our model is that we mainly rely on the bought items to calculate the similarity. However, there are many purchased items which are not liked by the buyers and they never play them again. They will not like the recommendation list which is generated based on all items they bought, since it might be ones related to un-liked games. We need to consider the likeness of the games by users to find the similar games to their most personal liked games instead of to all purchased ones, so that they have a higher probability to buy the recommended games. The third concern we have is that the TF-IDF algorithm we implemented to extract features from the review text is not improving the accuracy of our prediction. We hold the skepticism that other kinds of models are more appropriate to serve to incorporate reviews into prediction calculation mechanism.

4 RELATED LITERATURE

4.1 Previous Work

The data set we choose is from the *Steam* game which is a digital distribution platform for purchasing and playing video games. We adopt 82276 user accounts information, which consist their reviews, bought items, play hours, and recommends, to train the model to predict whether certain users bought certain items. We use the data set to calculate item similarities and user similarities respectively.

The similar data sets are *Steam Database*[2] provided by a third-party. In this data set, it specifies the number of players for each game and how long they play on this game in both real time and history. It ranked all games based on total played hours. Also, there is *Steam game data API*[3] which enables developers to access user inventory. Some previous open source *Steam* recommenders online use such kind of data to generate recommendations based on a user's top games which ranks playtime on all purchased games by this user and then suggest the games in the same categories of the most played games by this particular user[4]. Another project recommends for particular users based on collaborative Filtering algorithm which is a method of making predictions about the preferences of a user by collecting interests information from many users[3]. They formulated this CF problem as a learning problem and approach it using matrix factorization rendering by Alternative Least Square. The data used by this algorithm includes games' information such as price overview and players' information consisting users' owned games, users' friend list, and users' recently played games.[5]

The official recommender algorithm used by *Steam* considers different groups' demands so that it will use different algorithms respectively. From the current point of view, this algorithm and recommendation tips are based on whether you have played a game of the same type, the evaluation of other players, the association between your friends and the game (how many people have, how many people want, how many have given praise/bad reviews and whether you have viewed the relevant tags of the game. Its goal is to give gamers who like a certain gameplay a chance to learn more about similar works. It also asks the user to classify its recommendation of game is relevant to this user, and *Steam* improves recommender system algorithm based on the users inputs.

However, many players found *Steam* recommender not helpful for them to discover their interested games on this platform. The reason is that the *Steam* algorithm is inclined to recommend games with good sales or high attention. For those who play very few, games with high ratings are rarely exposed. *Steam 250*, a new recommender system, developed by a group of players used a new grading mechanism to score each game, and list the top 250 universal games for every user.[6] The algorithm is mainly based on user reviews. It takes two inputs: the number of votes and the approval rating which is the percentage of positive votes. Combining those with a fine tuned weight, each game has a score to be evaluated in the ranking. They focus on the maintenance of the purity of data by ignoring other complex features. This algorithm gains a wide popularity in the game players community.

4.2 State-of-the-art Methods

Many works in recommender system looks into how to use the neural net such as RNN, CNN in the recommender system application. Research works explore how the Content-Based recommendation and Collaborative Filtering using the deep neural net. The field of deep learning in recommender system is flourishing.

[2]https://steamdb.info/
[3]https://developer.valvesoftware.com/wiki/Steam_Web_API
[4]http://cjqian.github.io/projects/steam_rec/index.html?76561198124635511
[5]https://github.com/huntingzhu/Steam_Recommendation_System
[6]https://steam250.com/

The revolutionizing deep learning is able to overcome obstacles of conventional models by capturing the non-linear and non-trivial user-item relationships and represent more complex abstractions in the high layers. Moreover, it catches the intricate relationship of data itself. [4]

For the similar data set, previous researches have shown that CNNs can be applied to vanilla collaborative filtering. He et al.[5] uses outer product instead of dot product to model the user item interaction patterns. CNN are applied over the result of outer product and could capture the high-order correlations among embedding dimensions which the traditional model fails.

The integration of recommendation system into video game to maximize purchases and help users to get more personalized recommendations in modern free-to-play titles. *A Machine-Learning Item Recommendation System for Video Games* [6] experiments on DNN and ERT separately to predict whether users will buy some items in the future. The result show that both models show similar patterns and a relatively high accuracy.

4.3 Comparisons with our own models

Deep learning is a holy grail which we have not applied those techniques in our model. However, our model that has the best performance is the one with the simpliest structure which takes the least information of the data. It reasonates with the *Steam 250* which only counts the users' votes and their positive percentage in the votes. When we tried a more advanced model, the accuracy goes down. Also, another research work supposes that using additional features on similar models of ours will not improve the *Steam* recommender system. Their results showed no sign of improvement over the analysis containing only playtime.[7] We suspect that the structure in the data set we are dealing with is unable to perform well on our built advanced models so that a deep neural net might be needed to handle the situation.

5 CONCLUSIONS

Among the different models we developed, *Similarity Between this Game and Other Games Purchased by this User* model works the best, and *Purchase History of Other Similar Users* model also works well. The reason behind our decision is that, users tend to purchase more of the similar games due to similar interests, game advertisement in the same game category, bundle purchases of similar games. Users who like similar games (the collection of games in the same or closely-related categories) tend to purchase similar (and very likely) popular games.

Since this model is mostly tuned based on a similarity threshold and a count threshold (Refer back to Section 3.2 Optimization for these two threshold concepts, and ref back to Section 3.3 Other Models for what features is helpful), a closer look at what these two optimal thresholds mean will be beneficial. First off, a similarity threshold of 0.21 seems relatively high in terms of a threshold for Jaccard Similarity. This could means that users do follow our assumptions that they tend to stay with a certain set of similar games or closely-related genres of game that they will likely purchase. As for a count threshold of 0.0015, this surprisingly low threshold actually can be explained with our exploratory analysis from Section 1. We learned from the exploratory analysis that the ratio between the number of users and the number of games is at around 8 to 1 and the average number of games a user owns is around 58. Since that the number of games is relatively large comparing to the number of games a user owns, this could potentially indicate that a user does not have to own a large proportion of similar games to have a tendency to purchase another similar game.

Most of the other models that we tried have not been successful in exceeding our top model in terms of the test data accuracy. Total playtime, as the exploratory analysis also indicates, does not have a strong correlation with the number of games owned. The low accuracy on this model shows that a large total time spent on *Steam* games does not direct to a tendency to buy more games, potentially because they will likely be attached to a certain game and does not purchase and play other games. (For more models, refer back to 3.3 Other models.)

We also tried to combine features into the same model in the hope that they can yield better predictive results. However, several promising attempts have all failed to improve on the best performance of our single-feature model. Throughout multiple projects during this course, an interesting conclusion might be that the solution with the fewest assumption tends to yield the best results, which is exactly what Occam's razor recites.

Given the limitations of the data set we use, we have not used more advanced features of the data such as the friends list of the user, wishlists, and the categories of the games. In the future, with data containing more various entries, using the state-of-the-arts methods in the recommender system, we believe that the accuracy of prediction can be improved much more in a more personalized dimension. By improving the recommendations, *Steam* can boost its sales greatly and users can expand their libraries of games more efficiently.

6 REFERENCES

[1] Wan M, Mcauley J. Item recommendation on monotonic behavior chains. Proceedings of the 12th ACM Conference on Recommender Systems - RecSys 18. 2018. doi:10.1145/3240323.3240369

[2] Pathak A, Gupta K, Mcauley J. Generating and Personalizing Bundle Recommendations on Steam. Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval - SIGIR 17. 2017. doi:10.1145/3077136.3080724

[3] Xiaoyuan Su and Taghi M. Khoshgoftaar, fiA Survey of Collaborative Filtering Techniques,fi Advances in Artificial Intelligence, vol. 2009, Article ID 421425, 19 pages, 2009. https://doi.org/10.1155/2009/421425.

[4] Mu, R., Zeng, X., Han, L. (2018). A Survey of Recommender Systems Based on Deep Learning. IEEE Access, 1-1. doi:10.1109/access.2018.2880197

[5] Xiangnan He, Xiaoyu Du, Xiang Wang, Feng Tian, Jinhui Tang, and Tat-Seng Chua. 2018. Outer Product-based Neural Collaborative Filtering. (2018).

[6] Bertens, P., Guitart, A., Chen, P. P., Perianez, A. (2018). A Machine-Learning Item Recommendation System for Video Games. 2018 IEEE Conference on Computational Intelligence and Games (CIG). doi:10.1109/cig.2018.8490456

[7] Improving Steam System Performance: A Sourcebook for Industry, Second Edition (Book) (Revised). (2012). doi:10.2172/1094887