Steam Game Recommendations for Australian Users

Zhengyu Jiang
University of California, San Diego
jzhengyu@ucsd.edu

Yipeng Xiao
University of California, San Diego
yix017@ucsd.com

Ming Ki Toby Cheng
University of California, San Diego
mkc010@ucsd.edu

I Introduction

Steam is a popular video game distribution platform developed by Valve, one of the largest US based game distribution companies. This dataset covers All Steam Game review data, All Steam Game Metadata, Steam Australian user-game data, Australian user review data, and game metadata.

In this analysis, we will attempt to draw general insights about the data through exploratory analysis, create a predictive model, and compare our model and analysis to other published literature regarding this dataset.

II Literature Review

The dataset we use comes from https://store.steampowered.com/ and contains features such as users' purchase information, play times of the games and reviews they write. Some other studies choose the same dataset as ours, but crack the problem from different angles.

In *Item Recommendation on Monotonic Behavior Chains* [1], the author stated that explicit and implicit feedback, though has been studied for many years, often appears only in isolated fields. However, from the observations of real-life systems, which contain both explicit (click and purchases) and implicit (reviews and recommendations) information, it's only natural to raise the question that, can we leverage implicit signals to help predict explicit signals? The author referred to a serial user behavior as 'monotonic behavior chain' (ie. from click, purchase to review and recommend), and tries to build a model to detect the dependencies between different stages. In that case, different types of interactions could be unified and then lead to decision-making simulation of users.

The recommendation problem was expressed as to predict users' responses for new items given their historical behavioral chains. For the preliminary studies, the thesis provided three learning strategies to proceed with.

- *Learning preferences independent of stages* ignored the relations between stages, with examples as logistic regression and latent factor model.
- *Learning preferences jointly on different stages* focused on estimation within each stage and joined stages by adding marginal probability.
- *Learning preferences conditioned on previous stages* changed marginal probability into conditional probability assuming the latter stage was conditioned on the execution of former stage.

The last strategy best captured the behavior chain theory but faced data scarcity, therefore it was difficult to reflect the noisy data in subsequent stages. The author developed chainRec with two techniques Monotonic Scoring Function and Edgewise Optimization Criterion to solve this problem. Instead of conditioning on all previous interactions, the new algorithm tried to identify intrinsic and derived intentions, selecting only critical features at each stage.

Then the article experimented with three groups of models (independent, slicewise,

chainRec) on *Steam* dataset, and chainRec outperformed the other two in the whole behavioral chain, which proved that leveraging behavioral chain enhanced predictions on stage responses.

The method this article took is different from ours in that it identifies user behavioral stages and takes advantage of inner-stage relations. In terms of behavioral chain derived from *Steam* dataset (purchase - play - review - recommend), our popularity and similarity model focus on purchase stage only. And for our logistic regression model, we throw in features coming from the first three stages but treat them equally without specifying the causal relations. Technically, the thesis reached a more advanced level in approaching the situation.

In *Generating and Personalizing Bundle Recommendations on Steam* [2], the author uses a similar dataset as ours, only adding bundle purchase information to the traditional recommendation data. This article develops a bundle BPR model to recommend bundles, and greedy algorithm to create new bundles.

III Dataset and Analysis

The All Steam Games Metadata contains various information regarding all the games on Steam. The possible information is the game id, game url, reviews url, title, publisher, developer, discount price, price, genres, tags, release date, specs, and early access. However, not every entry has every one of these variables which means there will be omitted games when trying to summarize the whole dataset. There are 32135 different games, 339 tags, 22 genres. 6.45% of the games are early access games.

In **Figure 1** below we see the top 10 game publishers on Steam, as in they produce the most games. Ubisoft and Dovetail Games - Trains lead in this aspect by a large margin.

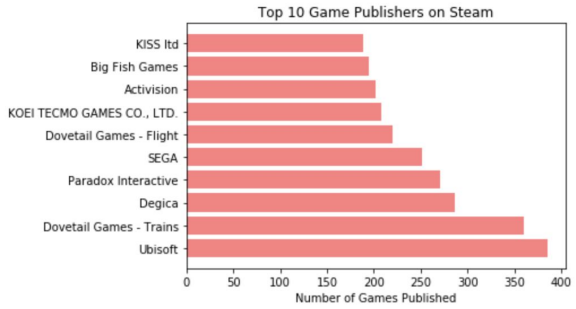

Figure 1: Top 10 Game Publishers on Steam

Next we looked at the most popular Game genres. **Figure 2** shows the Top 10 Genres on Steam. It shows that Indie shows up the most in games with Action following second. However, it's important to keep in mind that every game can have a multitude of genres. Indie only tells us that it was created by a small group of individuals or not a large publisher. It does not really tell us anything about the game. It would be more accurate to say that Action is the most popular genre.

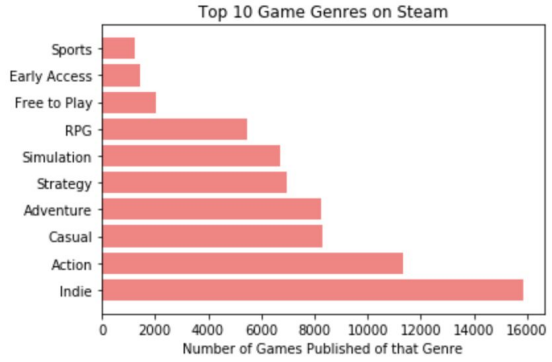

Figure 2: Top 10 Game Genres on Steam

The third thing we examined were the distribution of prices offered on Steam. Initially, we simply looked at all the possible prices and their distribution. However, this led to unforeseen problems since there are some games with an extremely high listed price. The maximum price is $995. Additionally, many games that are free were not listed as $0. Instead they were listed as some variation of 'Free' with some being simply 'Free' and others being 'Free To Play'. As such we had to clean the data and replace any prices with a

string that contains 'Free' to be replaced by $0.00. Despite this, there were still 10 strings in the prices, however there was no pattern to these strings. So we dropped these strings since it is a really small amount of the total number of prices (around 10 out of 30758 total prices). To get a good visualization of prices we limit the maximum of $60 and minimum of $0.00. A boxplot of this distribution is shown in **Figure 3**.

Figure 3: Boxplot of Distribution of Prices

The average of all these prices was $8.87 with quartile 1 at $2.99 and quartile 3 at $9.99.

The Steam Game review data contains the the following per review: user id, url, if the review is voted funny, posting date, last edited date, item id of the review, helpfulness of review, overall recommendation, and review text. Here is a breakdown of the variables in the data.A more detailed description can be found in **Table 1**.

User_id	Unique ID of a User
Item_id	Unique ID if a Game
Playtime_forever	Total playtime of a User given a specific Game
Playtime_2weeks	Recent 2-week playtime of a User given a specific Game
User_Review	Review of a Game given by a User
Publisher	The Publisher of a given Game
Genres	The Type of a given Game
Tags	Tags of a given Game
Price	The Price of a given Game

Table 1: Breakdown of Variables in All Data

The Australian Steam Video User and Item Data contains user-game data where each user and all the games they own and the playtime per game. They can be broken down into 800,000 user-game pairs. The Australian Steam user review data contains every user and all the reviews they're written and the posting time. The Basic statistics regarding these are shown in **Table 2**.

Users	62423
Total Games	9245
Total Reviews	59305
Avg Total Playtime Per User	8115
Avg Number of Games Bought Per User	8.22
Avg Number of Reviews Given Per User	0.58

Table 2: Statistics of Australian Steam Video

Then we looked at the most played games by Australian users. We defined this by the total number of hours of that game played by the users. The top 10 most played games are shown in **Figure 4**.

Figure 4: Top 10 Most Played Games by Australian Steam Users

Next we looked at the most owned games by Australian Steam Users. We went through every user's library and added a count for every game they had. **Figure 5** shows the top 10 most owned games by Australian Steam Users.

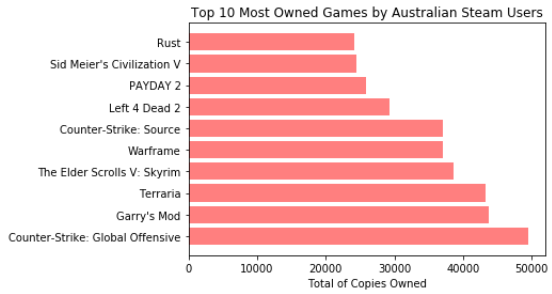

Figure 5: Top 10 Most Owned Games by Australian Steam Users

Finally we looked at the top 50 most popular tags in games played by Australian users. A word cloud is shown below in **Figure 6.** We found that the tags are mostly consistent with our previous findings. Multiplayer, action, and shooter are the top 3 tags and that is consistent with the most popular game (Counter-Strike: Global Offensive).

Figure 6: Top 50 Most Popular Tags in Games played by Australian Steam Users

IV Problem Formulation

From examining the data set, we could run a model to predict whether or not the user would buy the game. We have data for all the games owned by every user, the reviews written by the users, and the playtime for each game they own. We can create models to predict if they would buy other games based on their library and data. For baseline models we can predict whether or not they will buy based on the popularity of the game (Similar to what was done in Homework 3). Then we can consider similarity metrics such as Jaccard Similarity as well as combining Jaccard Similarity and Popularity. Finally we will use logistic regression on the different aspects of the user data to compare its accuracy to the baseline models.

V Predictive Analysis

A. Baseline Models

First we took our user-item-playtime data and filled it into a dataframe. Then we split the data into Training, Validation and Test Data in a 60%, 20%, 20% ratio. Using training data, we formed 2 lists of dictionaries, one of all the users who have bought that game, one for all the games bought by that user. Additionally we created a set of all the games in the dataset. We formed negative user-game samples by taking the difference between the set of all games and the set of games the user has played and choosing a random game in that difference to pair to the user.

To create the first baseline model based solely on popularity we first counted how many times a game showed up in the training set. We opt for this model out of intuition: if an item is popular then the possibility of it being purchased is higher. This model's strength lies in its simplicity, but is somewhat crude considering the prediction standard. For the model, for every user-game pair in validation, we predicted that the user would/has bought that game solely based on if that game was in the top 50% most popular games in the dataset. The resulting model accuracy came out to be 73.38%.

The second baseline model is a Jaccard similarity model. For every user-gameA pair in the validation data set, this model takes every gameB the user has already bought. For each of those games, it calculates the Jaccard similarity between the users who have bought the concerned gameA and the users who have bought gameB. However we had to take into

account that the user or game in the pair may not have shown up before in the training set. In this case we set that if either hasn't been seen simply return the mean Jaccard similarity value to be 0. After all the Jaccard similarities for the games of that user have been calculated, the model takes the mean of all these Jaccard similarities and if it is above a designated threshold, it predicts the user has bought this game. As before we looped through a range of thresholds to find the ideal threshold to give the highest validation accuracy. The ideal threshold is 0.004 with corresponding accuracy of 80.83%. A graph showing the different accuracy values is shown in **Figure 7**.

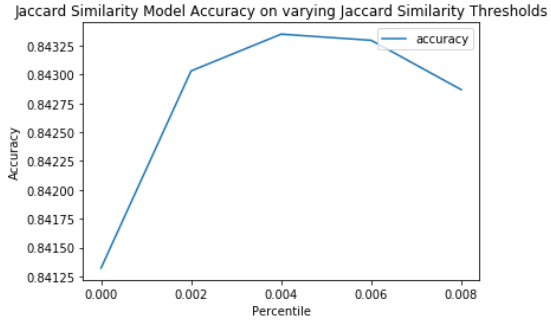

Figure 7: Jaccard Similarity Model Accuracy on varying Jaccard Similarity Thresholds

B. Improving on Baseline

First, to improve the popularity baseline, we modified the popularity threshold to the one that yielded the highest validation data accuracy. We checked this by looping through different thresholds to divide the total count by (from 60% to 100% of the most popular games). In other words, if the game was in the top 60% varying up to 100% of games, the model would return 1. **Figure 8** below shows the varying validation data accuracy based on varying thresholds. As shown, using this model, the maximum accuracy is with a popularity threshold at 0.85, with accuracy being 83.42%, which is much better than the pure popularity model accuracy of 73.38%.

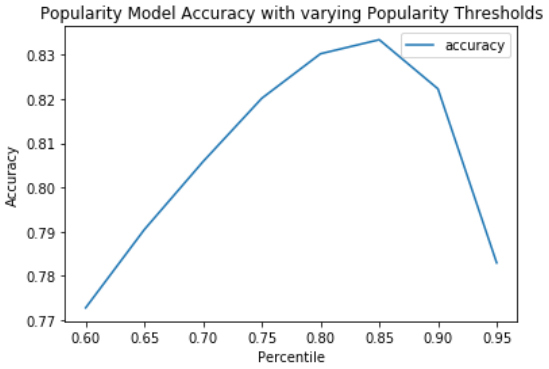

Figure 8: Popularity Model Accuracy with varying Popularity Thresholds

Both Popularity and Jaccard models have a decent accuracy on their own, but we could likely become even more accurate by combining. For this model we directly multiplied the game popularity count to the mean Jaccard similarity value. Similar to before, the model will predict the user has bought a given game if it passes a certain threshold. The ideal threshold turns out to be 0.16 with accuracy 84.4% as shown in the **Figure 9** below. This is the highest accuracy we've gotten so far by simply modifying similarity and popularity metrics. Next we will try to use logistic regression to beat this model.

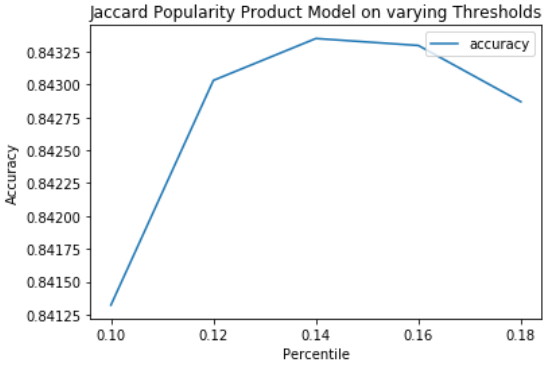

Figure 9: Jaccard Popularity Product Model on varying Thresholds

We choose the logistic regression model because what we predict here is whether users will purchase unobserved items in the future, which is a 0/1 prediction task. Logistic regression will return a probability and predict 1 if it exceeds 50%. For features used in

logistic regression, for every user-game pair, we took the user's total library playtime (playtime), the number of games they have in their library (n_item), the number of reviews they've written (n_review), game Jaccard similarity by comparing users (same as in baseline model, here referred to as avg_sim), and the raw popularity value of that game (popu). Similarly for Jaccard similarity we have to consider new users or items. For users who did not appear before, we assign the mean value of playtime, n_item, n_review and 0 for avg_sim. For items that did not appear before, we assigned it the average popularity value.

For the first logistic regression model we simply used these 5 features and fit the model using a training set. After many trials of different values of the inverse regularization value C, we found that value of 10 gave the best accuracy. This basic logistic regression model had an accuracy of 79.31%.

We also tried using a Support Vector Machine on our features. Support-vector machines are supervised learning models built for classification and regression analysis. The models operate by finding the dividing line, hyperplane or set of hyperplanes to achieve the result that data points nearest to the classifier have the largest distance to it. The resulting accuracy of the SVM model was 72.68% which is quite a bit lower than our first logistic regression model. As such, we carried out the remaining model improvements only on the logistic regression model.

As our first logistic regression was worse than our best baseline model (Jaccard), we tried to modify the features to improve accuracy. From our visualizations from before, the data showed to be right skewed in general. For example, most people have fewer games and write less reviews. As such, we tried to natural log all features other than Mean Jaccard Similarity to improve the model accuracy. However, when making these log transformation, we had to make sure to only log on non-zero values. After more trial and error of the regularization value, it was still best at value of 10. The natural log logistic regression model had a big improvement, validation accuracy jumped to 84.61%.

Although we have managed to build a more accurate model, the model was barely higher than the combined Jaccard and Popularity model which had an accuracy of 84.4%. We then considered that some of the features in the regression may have too high a value that is causing too big an impact on the regression. Although natural log would decrease this value, it may not be enough. Therefore, we took the base 10 log of playtime and popularity, which are very high value features, while keeping the other features natural log-d and Jaccard Similarity unchanged. The resulting accuracy on the validation set was much improved, at 90.25%. So our final logistic regression model uses the following features: log10(playtime), log(n_item), log(n_review), avg_sim, log10(popu).

VI Conclusion

Table 3 shown below is a summary of all our model performances. Our combined log Logistic Regression performed the best, 9.42% better than the best baseline (Jaccard).

Model	Accuracy
Half Popularity	73.38%
Jaccard Similarity(0.04)	80.83%
Popularity(0.85)	83.42%
Jaccard * Popularity(0.16)	84.40%
Simple Logistic Regression	79.31%
Simple SVM Model	72.68%
Natural log Logistic Regression	84.61%
Combined log Logistic Regression	90.25%

Table 3: Summary of Predictive Model Performance

The coefficients obtained from the best model were the following -2.404, -0.0488, 0.314, 0.101, 111.2, 2.097. The corresponding features are offset value, log10(playtime), log(n_item), log(n_review), avg_sim, log10(popu). Because all the variables are of different scale, it is hard to interpret each coefficient specifically. Instead we can look at the general positivity of the coefficients. In general, the higher overall playtime the player has, there is a larger negative effect on whether or not they will buy the game. This is somewhat logical as someone who is already spending many hours in their existing library is not as inclined towards buying new games. They are already getting plenty of enjoyment out of their existing library. The remaining variables are all positive and this also makes sense. The higher the number of items the player owns and the number of reviews they write, the more interested the user is in games. The higher the mean similarity, the more similar the game is to the user's current games. Finally, it makes sense that the more popular the game is, the more likely people would want to buy and play it.

Something that could be explored in the future would be the effect of review sentiment combined with sentiment to improve the recommendation further. Another interesting topic that could be explored the preference of the users in terms of genre/tags and the game's genre/tags. Additionally, we would be able to make a more accurate model or more widely applicable model if we had worldwide user-game data available to us.

REFERENCES

[1] Wan, M. and McAuley, J., 2018, September. Item Recommendation on Monotonic Behavior Chains. In *Proceedings of the 12th ACM Conference on Recommender Systems* (pp. 86-94). ACM.

[2] Pathak, A., Gupta, K. and McAuley, J., 2017, August. Generating and Personalizing Bundle Recommendations on Steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1073-1076). ACM.