Steam Games Recommender Based on Calculated Ratings

Michael Treacy
Undergraduate in Computer Science
University of Massachusetts Lowell
Lowell, USA
Michael_Treacy@student.uml.edu

Nathan Lamberson
Undergraduate in Computer Science
University of Massachusetts Lowell
Lowell, USA
Nathan_Lamberson@student.uml.edu

***Abstract*—Two models were created to recommend a list of video games for users in a Steam data set of gamer purchases and hours played. To ease the application of history-driven, or collaborative, filtering techniques, ratings were calculated from the hours data for model 1 [1]. The original hours were used in model 2. The results showed more meaningful recommendations in model 1. Suggestions included games of a similar genre to the users' already-owned games. Model 2 had low precision scores, but served a good foundation for model 1. Future work would see a greater removal of outliers in the set, the use of more product and demographic information when finding similar users and filtering recommendation results, as well as an exploration in the field of configuring more complete user sentiments from a greater variety of behavior.**

***Keywords*—steam, valve, video games, recommender system, collaborative filtering**

I. Introduction

Recommendation systems for online products rely on user data in order to customize the consumer's experience and help them find the items they are looking for. Depending on the information available, user sentiment is developed, and different algorithmic techniques can be applied to those sentiments to reach the goal of recommending. As researchers, our focus was to attempt this with data from the Valve Corporation's Steam video game marketplace. More specifically, we would take the data set's information on each player's purchase and game hours history, use that history to configure a form of game preference, and recommend new ones to individual players from those preferences.

A. Task Environment

The environment, or state space, of a learning agent is often difficult to handle. In our case, we were dealing with the highly variable world of games and players. Through its virtual sensors, the recommendation agent would basically collect game information and the behaviors of multiple users, make assumptions of what a specific player may like depending on the other players he or she is similar to, and produce the video games to recommend that user. This would all be to increase the likelihood that the gamer will find what he or she wants to buy. The recommended games, or games-feed, and computer display could be viewed as the main actuators.

B. Properties of the Task Environment

It seems reasonable to posit that the actual environment our agent would be deployed in after testing on this static data set could be described as partially observable, multiagent, stochastic, sequential, dynamic, continuous, and known [2]. The environment is partially observable because the agent's sensors does not give it access to every game in existence or the user's thoughts, cooperatively multiagent because both agent and player seek a better list of games, stochastic due to the software available always changing and gamer behavior not being completely determined by the agent, and sequential because each feed decision can affect later decisions. It is also dynamic as a result of new video games always popping up and the user continually changing his or her mind while the agent is at work updating, continuous because there is an ever-changing number of states and games-feed configurations, and known because the agent always holds the state of knowledge about how it can work with the games and players.

C. COMPONENTS OF THE AGENT

In terms of Russel and Norvig's descriptions of learning agent components, the critic here has standards for things like games bought and hours played, which the learning element and evaluation function use to decide whether the agent successfully predicts what customers want [2]. The learning element also gathers knowledge and provides changes to the performance element so that it, in turn, can manipulate the games-feed to better match with the user's preferences. Lastly, if this recommendation system were to be released, the problem generator would inform the performance element to try some random games to introduce to gamers in order to see how some slightly different feeds work with the players. Understanding our agent with these learning components helps in conceptualizing the more complex filtering algorithms needed in order to solve our problem of game recommendation.

II. Literature Review

As stated by researchers Mohamed, Khafagy, and Ibrahim, the development of a recommendation system depends on the use of popular techniques such as demographic, content-based, collaborative, and hybrid filtering [3]. Demographic data consists of general information about the user such as gender or location, while content-based filtering focuses more on aspects of the actual products being bought. On the other hand, the collaborative method involves things like making use of the purchase history of all customers and applying association rule mining in order to detect commonalities that will be helpful in predicting people's preferences [1]. Lastly, a hybrid approach mixes these filtering techniques. Looking at each of these focuses, it was clear that the collaborative method of taking into consideration user behavior would be the most useful to us [4].

That is why the main focus of our student project was to apply collaborative filtering to the collection of Steam user-and-product data we found. The Valve Corporation already has a recommendation system in place. Data ranging from a gamer's wish-listed items to his or her reviews is collected by the company to guess what type of games the individual may want. Due to our data set only containing purchase and play history and collaborative approaches often depending on the consumer's item sentiments, we sought to calculate ratings from the hours played in order to have some type of player preference. Helpful guides in scaling and filtering would be utilized. More specifically, min-max normalization would be implemented to reformat the hours into a player rating [5]. This would then be followed by filtering for item-based recommendations [6]. Finally, the user-based recommendation systems would be explored and developed. Each researcher created their own model for the user-based systems in order to provide a means of comparing and learning small deviations in collaborative filtering techniques. Model 1, primarily researched and coded by Michael Treacy, is reported first [7]. This model made use of the scaled data set. Next, model 2, mainly investigated and programmed by Nathan Lamberson, is covered [8]. This second approach did not utilize the scaled data and instead built a system around the original hours the players gave to their games. Having one model use the scaled data and another use the raw hours allowed for a good comparison opportunity of the impact of scaling. The ultimate hope was to create novel solutions to Valve's problem of finding the right games for the right consumers.

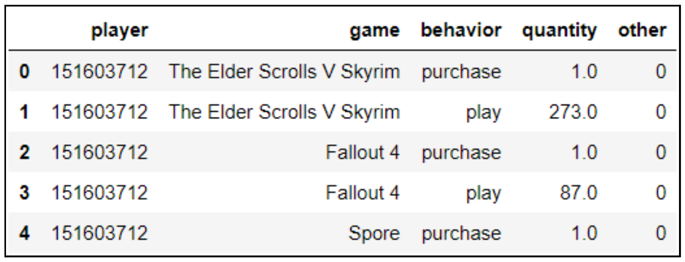
III. Methodology

A. CLEANING AND SCALING METHODS

The data set under question can be found on Kaggle, which is a site tailored for the machine learning and data science community [9]. It has a total of four columns consisting of 'user-id', 'game-title', 'behavior-name', and 'value', where the behaviors are 'play' and 'purchase' and their associated values, respectively, are how many hours have been given to a game and whether the game was bought [9]. We would begin by downloading the set, cleaning it, and converting it into a usable form inside the jupyter notebook framework using python libraries such as numpy and pandas [10]. These libraries are especially helpful in conducting mathematical operations on matrices.

When it came to our needs, it was clear that the data held some unnecessary information. This is evident from Table I. Columns labeled as 'behavior' and 'other' were not required. Similarly, the rows with a 'behavior' of purchase and 'quantity' of 1 did not say anything useful. As a result, they were removed. We also deemed it important to remove gamers with a game count that was less than 5. This was so that we would only run our recommendation algorithms on more active accounts.

TABLE I. ORIGINAL DATA SET



	player	game	behavior	quantity	other
0	151603712	The Elder Scrolls V Skyrim	purchase	1.0	0
1	151603712	The Elder Scrolls V Skyrim	play	273.0	0
2	151603712	Fallout 4	purchase	1.0	0
3	151603712	Fallout 4	play	87.0	0
4	151603712	Spore	purchase	1.0	0

In order to scale the hours so that they took the form of a game rating given by the player, we made use of the min-max normalization formula shown in Eqn. 1. As indicated by this formula, we created a 1-10 rating by taking each game a user had played, subtracting the minimum number of hours they played from the total hours the game under question was played, divided that by the difference between the maximum amount of hours they gamed and the minimum, multiplied the result by the difference between the desired maximum and minimum, and, finally, added the desired minimum. This meant that the game played the most would be given a score of 10, while the game played the least would be given a rank of 1. Everything in between would be scaled between those values.

$$v' = \frac{v - \min_A}{\max_A - \min_A}(new\_\max_A - new\_\min_A) + new\_\min_A$$

EQN. 1. MIN-MAX NORMALIZATION [5]

B. RECOMMENDATIONS BY GAME METHOD

Implementing the algorithm for a recommendation system that takes a game as an input and offers similar games was inspired by some of the ideas behind Salem Marafi's collaborative filtering tutorial [6]. It first demanded the creation of a players-by-games data frame. The values used to create this were the ratings. Basically, every player took on a row and every game had its own column. For those games not played by a user, a 0 rating was entered. Otherwise, their score of a game calculated from the hours they spent playing it made up the cell.

This was then followed by a games-by-games data frame being declared and filled by taking the transpose and dot product of the players-by-games frame. Lastly, the final step was to fill in a new recommendation data frame by looping once through the columns of the games-by-games frame and sorting the scores found from the transpose and dot product calculation. It was expected that the end result would be a system in which games of a similar genre to the inputted game would be recommended.

C. MODEL 1: RECOMMENDATIONS BY PLAYER METHOD

Finding games for an inputted user would be a bit more involved. Santhosh's online guide on recommendation systems was helpful in this pursuit [7]. For model 1, it began with the determination of those players who were the most similar to the gamer passed in. The main formula utilized for this purpose was Eqn. 2. Here, the cosine similarity calls for getting the passed-in player's game ratings, finding the sum of products between them and each of the other player's scores, and then dividing each of those results by the relevant product of their squares, sums, and roots.

$$\frac{\sum\limits_{i \in I_{xy}} r_{x,i} r_{y,i}}{\sqrt{\sum\limits_{i \in I_x} r_{x,i}^2} \sqrt{\sum\limits_{i \in I_y} r_{y,i}^2}}$$

EQN. 2. COSINE SIMILARITY [11]

Once that was completed, a players-by-games data frame of the most similar gamers was created with the added column of their cosine similarities placed at the end. Weighted averages filled the empty cells in the passed-in player's row so that the highest averages, or ratings, could be recommended. These game averages were calculated by taking each other player's rating for that game, multiplying it by their similarity score, adding the result to a running tally for the game, and then dividing it by the sum of the cosines for all players.

The final step was to create a recommendation list for the passed-in player and fill it with the unplayed games holding a weighted rating. This could be done by iterating over his or her game row and checking for values over 0. After doing this and gathering the games, they were sorted from greatest to least making the beginning hold the games that the recommendation system assumes the player would rank the highest if he or she was to play them. The hypothesis was that a player's already-owned greatest ranked games would be of the same genre of the highest rated recommended games.

D. MODEL 2: RECOMMENDATIONS BY PLAYER METHOD

For model 2, a technique based on a tutorial found on Kaggle was used to help develop an understanding of the basics of building a recommendation system [8]. The main approach was matrix factorization. Model 2 helped set the backbone for implementing model 1. Matrix factorization is a type of collaborative filtering that is commonly used in recommender systems. It creates two matrices, U and V, where U is a preference matrix where the values represent whether a user has purchased a game in a list of games and V is a confidence matrix where the values represent the assumption of a user's hours played on a game and if they enjoyed said game. This replaces the need for a given rating or review and allows us to use the user's hours as a guess on whether they would like certain games or not. The matrices, U and V, are combined to make the R matrix of user recommendations based on the preferences and confidences found in U and V.

In order to make these matrices, we need to have properly cleaned data. This is done by removing all instances in the 'behavior' column that contains the value 'purchased' since this data is irrelevant and only yields a play time of 1.0. After removing the 'purchased' rows where 'hours' are exactly 1.0, we are left with only playtime data. We can now make our preference matrix, U, where the rows are the users, and the columns are the games. If a user has played a certain game, a value of 1 is recorded where the row and column meet. Otherwise, there is a zero in its place and the user has not played the game. Now the confidence matrix contains the same rows and columns, however, instead of a 1 indicating whether a user has played a game or not, their total playtime is recorded where the rows and columns meet. If a user has not played a certain game, the value is a 0.

After getting this, we have to remove the information that won't be relevant. This means we can trim out users that own less than 10 games to reduce inactive users that would disrupt the data. This brings our list of users down from 12,393 to 2,189 (model 2 code cell 10), which is a much more manageable and impactful data set. Following this, we begin training the model with roughly 20% of the data. This means that each time we run the algorithm we will be working with a new set of users, either helping or hurting the model. Sometimes this leads to high precisions, while other times to low precisions.

Once this was completed, we simply had to add bias to the system and determine the games that have large playtime due to never ending possibilities and those with set playtimes to ensure playtime accurately represents enjoyment. The former would include titles such as Counter Strike and Dota 2, while the latter include any story-focused games. This is accomplished by concatenating the preference matrix and a vector of the total user values and 1. We repeat this step for the confidence matrix and a vector of the total games and 1. These two biases are then matrix multiplied and we get our predicted preferences for users and games.

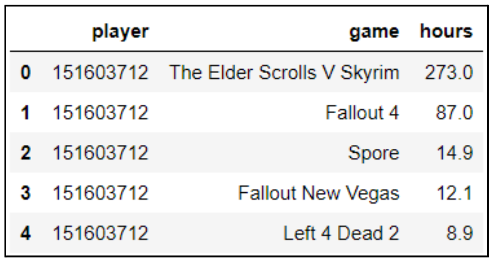
Lastly, we calculate the top k-precision when training the data, which takes the recommended games for a user, finds the top recommendation, and appends it to a list to find the top k games for the user. We would repeat this training step 100 times to continue to ensure the top recommended games were being accurately recommended across all iterations. Now the data is trained and we can select users from the users we sliced in an earlier step and see what games the model predicts they would like compared to what they have actually purchased. It was assumed they would be of a similar genre.

IV. Results

A. CLEANING AND SCALING RESULTS

The cleaned data consisted of only a 'player', 'game', and 'hours' column. It can be viewed in Table II. With the removal of the 'behavior' and 'other' columns, as well as the rows with a 'behavior' of purchase and 'quantity' of 1, we had a more straightforward data set to work with. It was also the case that the total number of players and games went down from 11,350 to 2,436 and 3,600 to 3,544, respectively, after accounts with less than 5 games in their play library were removed.

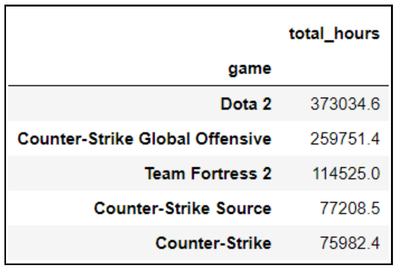
TABLE II. CLEANED DATA SET



	player	game	hours
0	151603712	The Elder Scrolls V Skyrim	273.0
1	151603712	Fallout 4	87.0
2	151603712	Spore	14.9
3	151603712	Fallout New Vegas	12.1
4	151603712	Left 4 Dead 2	8.9

This simplified set created opportunities to better explore it in a variety of ways. For example, we found that Dota 2 was the most played game with a 'total_hours' of 373,034.6. The rest of the top 5 most-played games are displayed in Table III. It was also revealed that the mean hours games are generally played is roughly 40.58.

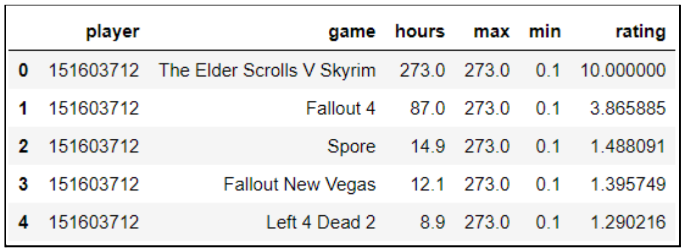
TABLE III. TOTAL HOURS BY GAME



	total_hours
game	
Dota 2	373034.6
Counter-Strike Global Offensive	259751.4
Team Fortress 2	114525.0
Counter-Strike Source	77208.5
Counter-Strike	75982.4

Determining the maximum and minimum for each player, along with his or her min-max normalized ratings for each played game, resulted in a data set of the form found in Table IV. This table shows that The Elder Scrolls V Skyrim was player 151603712's most played game. It's hours are therefore the value of the maximum and that game has been given a rating of 10 out of 10.

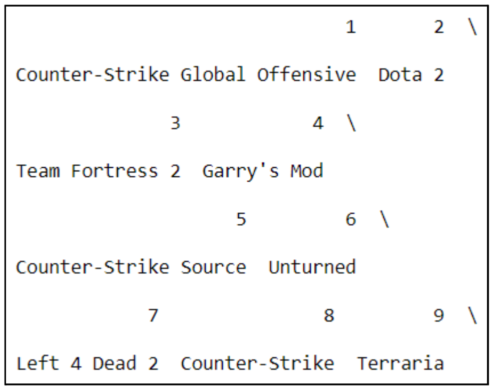
TABLE IV. CALCULATED RATINGS



	player	game	hours	max	min	rating
0	151603712	The Elder Scrolls V Skyrim	273.0	273.0	0.1	10.000000
1	151603712	Fallout 4	87.0	273.0	0.1	3.865885
2	151603712	Spore	14.9	273.0	0.1	1.488091
3	151603712	Fallout New Vegas	12.1	273.0	0.1	1.395749
4	151603712	Left 4 Dead 2	8.9	273.0	0.1	1.290216

B. RECOMMENDATIONS BY GAME RESULTS

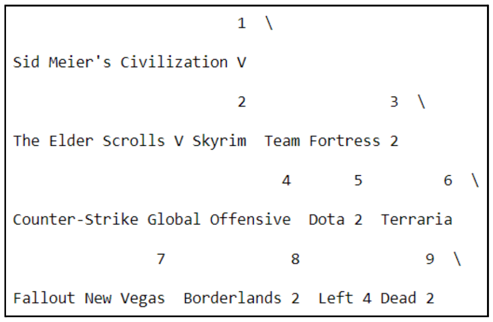
The recommendation system that would recommend video games based on the game passed in successfully listed out the desired number of software titles. Fig. 1 shows the output after inputting the first-person shooter Counter-Strike Global Offensive. As it can be seen, players who like that game should also try out games like Dota 2, Team Fortress 2, and Garry's Mod.



```
                                  1         2  \
Counter-Strike Global Offensive   Dota 2

               3                4  \
Team Fortress 2  Garry's Mod

                      5          6  \
Counter-Strike Source  Unturned

              7                8          9  \
Left 4 Dead 2  Counter-Strike  Terraria
```

FIG. 1. TOP 8 RECOMMENDED GAMES FOR CSGO PLAYERS

Similarly, Fig. 2 shows the results given when the system is fed the Sid Meier's Civilization V title. The Civilization games are more strategy-oriented. Gamers who enjoy that type of game may also find The Elder Scrolls V Skyrim, Team Fortress 2, and Counter-Strike Global Offensive worth investing some time in.

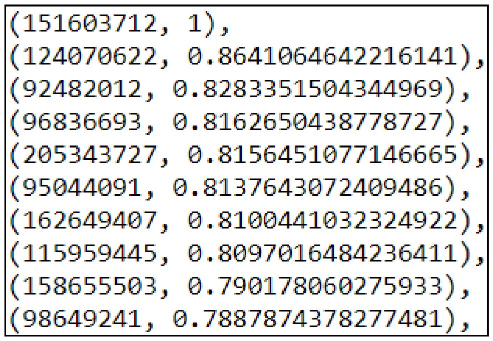


```
                          1  \
Sid Meier's Civilization V

                          2                3  \
The Elder Scrolls V Skyrim  Team Fortress 2

                                4       5         6  \
Counter-Strike Global Offensive  Dota 2  Terraria

                  7              8             9  \
Fallout New Vegas  Borderlands 2  Left 4 Dead 2
```

FIG. 2. TOP 8 RECOMMENDED GAMES FOR CIVILIZATION PLAYERS

C. MODEL 1: RECOMMENDATIONS BY PLAYER RESULTS

When it came to the model 1 results for player 151603712, the most similar gamers all had fairly close cosine similarities. This was typical for any user passed in. A total of 9, excluding the passed in player, are displayed in Fig. 3. The values shown range roughly from 0.86 to 0.79.

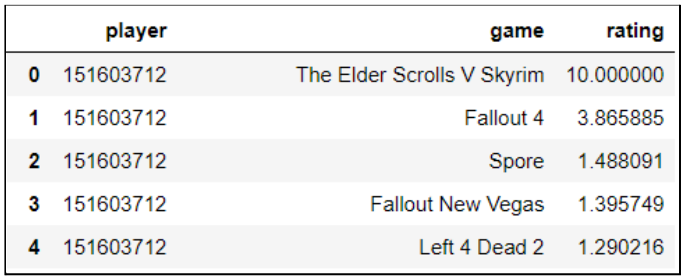


```
(151603712, 1),
(124070622, 0.8641064642216141),
(92482012, 0.8283351504344969),
(96836693, 0.8162650438778727),
(205343727, 0.8156451077146665),
(95044091, 0.8137643072409486),
(162649407, 0.8100441032324922),
(115959445, 0.8097016484236411),
(158655503, 0.790178060275933),
(98649241, 0.7887874378277481),
```

FIG. 3. TOP 9 SIMILAR GAMERS TO PLAYER 151603712

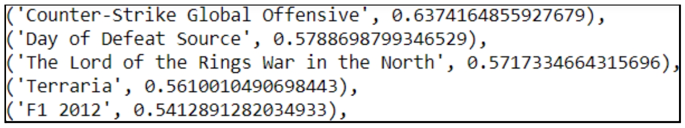
Some games already owned by player 151603712 can be viewed in Table V. The Elder Scrolls V Skyrim holds a rating of 10, while Fallout 4 approaches a score of 4. The rest of his or her games do not exceed 1.5.

TABLE V. RATINGS OF BOUGHT GAMES



	player	game	rating
0	151603712	The Elder Scrolls V Skyrim	10.000000
1	151603712	Fallout 4	3.865885
2	151603712	Spore	1.488091
3	151603712	Fallout New Vegas	1.395749
4	151603712	Left 4 Dead 2	1.290216

Fig. 4 are the top games the model 1 recommendation system would suggest to this player. They include Counter-Strike Global Offensive, Day of Defeat Source, The Lord of the Rings War in the North, Terraria, and F1 2012. The second number in each tuple is the weighted average, or rating, expected by the player.



```
('Counter-Strike Global Offensive', 0.6374164855927679),
('Day of Defeat Source', 0.5788698799346529),
('The Lord of the Rings War in the North', 0.5717334664315696),
('Terraria', 0.5610010490698443),
('F1 2012', 0.5412891282034933),
```

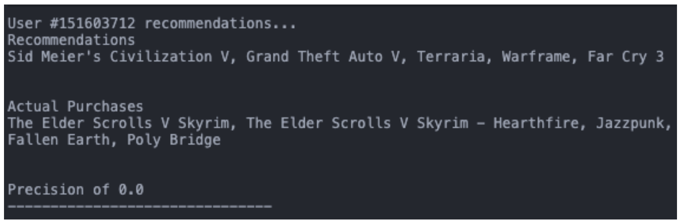
FIG. 4. TOP 5 RECOMMENDED GAMES FOR PLAYER 151603712

Results for another player, 44153929, come mostly from lowly rated games. For example, this player's top 5 most played games included Counter-Strike, Counter-Strike Condition Zero, Dota 2, theHunter, and Modular Combat, but their associated ratings are around 10, 1.02, 1.01, 1.01, and 1. Some of the recommended games for this user were Counter-Strike Global Offensive, Team Fortress 2, Counter-Strike Source, Call of Duty Modern Warfare 2 - Multiplayer, and Call of Duty Black Ops - Multiplayer. Their calculated ratings were roughly 1.10, 0.90, 0.77, 0.65, and 0.55.

A third player included user 110238251. Their highest-rated owned games were Empire Total War, Napoleon Total War, Total War ROME II - Emperor Edition, Total War ATTILA, and Mars War Logs, with scaled scores of 10, 9.11, 4.76, 3.74, and 1, respectively. Found spread out among the player's recommendation list was Sid Meier's Civilization V, Age of Empires II HD Edition, Kerbal Space Program, Total War SHOGUN 2, and Sid Meier's Civilization Beyond Earth. All had ratings below 1.5.

D. MODEL 2: RECOMMENDATIONS BY PLAYER RESULTS

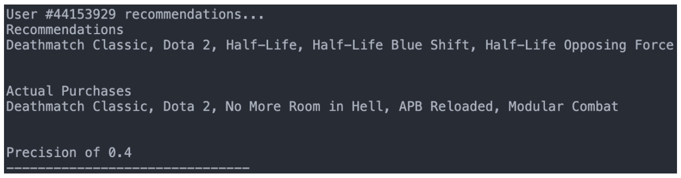
As for model 2, the recommended games for player 151603712 yielded a 0.0 precision, making no recommendations that matched the user's actual purchases. If we were to compare genres, we would have some close actual purchases, but this is not the information we are looking at. We see this detailed in the results shown in Fig 5.



```
User #151603712 recommendations...
Recommendations
Sid Meier's Civilization V, Grand Theft Auto V, Terraria, Warframe, Far Cry 3


Actual Purchases
The Elder Scrolls V Skyrim, The Elder Scrolls V Skyrim - Hearthfire, Jazzpunk,
Fallen Earth, Poly Bridge


Precision of 0.0
--------------------------------
```

FIG. 5. RECOMMENDED GAMES FOR PLAYER 151603712 FOR MODEL 2

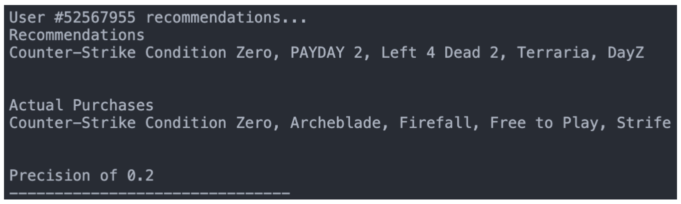
This trend proves to be true for most users, where the recommendations often yield a precision of 0.0 to a user's actual purchases. We do have occasional results that make some accurate recommendations, ranging between 0.2 and 0.4, but these are often far and few between. We can view this happening with user 44153929 in Fig. 6 below.



```
User #44153929 recommendations...
Recommendations
Deathmatch Classic, Dota 2, Half-Life, Half-Life Blue Shift, Half-Life Opposing Force


Actual Purchases
Deathmatch Classic, Dota 2, No More Room in Hell, APB Reloaded, Modular Combat


Precision of 0.4
--------------------------------
```

FIG. 6. RECOMMENDED GAMES FOR PLAYER 44153929 FOR MODEL 2

Additionally, we get some results of 0.2 precision. These can be seen in Fig. 7 below.



```
User #52567955 recommendations...
Recommendations
Counter-Strike Condition Zero, PAYDAY 2, Left 4 Dead 2, Terraria, DayZ

Actual Purchases
Counter-Strike Condition Zero, Archeblade, Firefall, Free to Play, Strife

Precision of 0.2
------------------------------
```

FIG. 7. RECOMMENDED GAMES FOR PLAYER 52567955 FOR MODEL 2

V. DISCUSSIONS

A. CLEANING AND SCALING DISCUSSION

One interesting piece of information that was found after cleaning the data set and checking the total number of hours each game has been played was that the top 5 most played games are all games made by the Valve Corporation. These are located in Table III. This brings to question whether Valve may recommend their own games more than others within their Steam marketplace.

Another thing to note when it came to the results of cleaning and scaling the data was the problem that occurs when a player has sunk a very large amount of time into 1 game and only a small, but more reasonable, amount of time in others. Table IV shows that the player under question played The Elder Scrolls V Skyrim for 273 hours, while the rest of his or her games have not even broken the 100-hour mark. Some games can simply be beaten in a small chunk of time. Is the fact that they weren't played as much as a sandbox game designed for exploration mean they deserve such a low score? It may be the case that a better way of calculating a rating ought to be configured or that such users should just be removed.

B. RECOMMENDATIONS BY GAME DISCUSSION

The item-based system, which was passed a game, could produce some meaningful results. More specifically, if a first-person shooter was given, the system would tend to output other first-person shooter games. As shown in Fig. 1, inputting Counter-Strike Global Offensive outputted other shooters like Team Fortress 2, Garry's Mod, Counter-Strike Source, Unturned, Left 4 Dead 2, and Counter-Strike.

On the other hand, it could also give some questionable results. For example, passing in Sid Meier's Civilization V, which is a strategy game, did not cause the recommender to produce any other obviously related strategy games. This is clear from the results displayed in Fig. 2. Reflecting on these contradictory types of output, it seems obvious that defining game similarity by whether or not 2 games fall into the same genre is a naive outlook. Video games ought to be compared in a multitude of ways such as story, art styles, and leveling systems.

C. MODEL 1: RECOMMENDATIONS BY PLAYER DISCUSSION

The first model for recommending games to a specific player outputted some high correlations when it came to the finding of similar gamers. Each of the users in Fig. 3 were above 0.75. This was a good sign and pretty common when different players were fed into the recommendation system.

One problem with model 1 was the very low calculated ratings. Players 151603712, 44153929, and 110238251 all had this issue. It seems possible that there are just too many games and the ratings need further tuning. As stated before, a player's most played game can throw off the rest of his or her formula-deriven ratings for lesser played games. Future work would see the removal of those outliers, as well as any other kinds that are determined.

Despite the shortcomings of the low ratings, some of the games suggested in Fig. 4 do make quite a bit of sense. Fallout 4, Fallout New Vegas, and Left 4 Dead 2, which player 151603712 already owned, are all first-person shooter games, while The Elder Scrolls V Skyrim is a fantasy role-playing game. The recommended games list holds Counter-Strike Global Offensive, Day of Defeat Source, and The Lord of the Rings War in the North at the top with the highest calculated ratings. This is 2 first-person shooter games and 1 fantasy role-playing game. Similarly, user 44153929 mostly had first-person shooter games and was recommended more of them. Lastly, 110238251 owned 4 out of 5 Total War games for their top rated games and a number of related strategy games were suggested to them. It is important to state that the strategy games were not all found at the top of the recommendation list, but they were included in it. The main point is that the suggestions do have similarities with the users' already-owned games. However, it needs to be considered that game similarity can be measured by more than just genre.

D. Model 2: Recommendations by Player Discussion

With model 2, the precision of the recommendations for video games were quite low when trying to recommend 5 games and then comparing them to the user's actual purchases. As indicated in Fig. 5 of the results section, we saw precisions of 0.0. Occasionally, we would witness better precisions in the range of 0.2 to 0.4 (See Fig. 6 & Fig. 7) where the recommendation system would recommend 1 to 2 games out of 5 and match them correctly with the user's actual purchases.

We have some theories for why these precisions are so low. First off, the slice of users being taken to train the data is too small, resulting in higher loss of information that could be used to make the recommendations more accurate. Second, simply using hours without any scaling does not offer enough to accurately predict the users' preferences for games.

For theory 1, this could be tested by simply changing the size of the slice of users from 20% to something larger. This, of course, would increase the time to execute all operations. Due to this fact, it was not something we got around to completely testing. However, it would be a good avenue for future exploration in order to determine if this current model could be improved. As for theory 2, you can see the scaling of data considered in the implementation of model 1, where a system is set in place to modify the hours into a rating system since there were no user ratings given for any of the games played. This allows for a more accurate system that can make a guess on the users' exact preference for certain games based on their calculated scores and the ratings of other players who are similar to them. The result is weighted averages for unplayed software, which can be suggested. Even though model 2 did not perform as well as model 1, it was a good foundation that would lead to an opening of our eyes to the ways in which we can optimize our data so that our model performs better.

E. Comparison of Models Discussion

As indicated above, model 1 outperformed model 2, but the former would not have been developed without the knowledge gained in implementing the latter. Model 1 would suggest more games of a similar genre to the player's already-owned games, while model 2's similarities between games played and those recommended were less clear. After reflection, though, we cannot stress enough that game similarity can probably be measured in a multitude of ways. Aspects of a video game such as its story, art style, leveling system, and the like should be considered along with genre in order to determine how connected different games are. It may be the case that our models performed worse or better than each other in ways we have not been able to see as a result of the one-dimensional measuring of their success primarily by how similar the genre of a player's owned games are to his or her recommended games.

VI. Conclusion

In conclusion, we have been able to successfully build two working recommendation models for video game players using the Valve Corporation's Steam marketplace. Model 1 was built with a scaled version of the data set and yielded seemingly high effectiveness when it came to recommendation. In short, it was able to successfully map users' hours to recommendations of unplayed games in a similar genre. Model 2 also recommended games, but directly from the playtime. The result, at least in terms of our way in which we were measuring similarity between games played and those suggested, was a less efficient recommendation system. Accuracy and precision, often a 0.0, were fairly low when trying to recommend just 5 games to a user. Occasionally, it would guess around 1 to 2 games with a higher precision of 0.2 to 0.4. This seems to be due to using unscaled data. Model 2, however, did set the baseline on how to go about improving the model to make it more accurate, which we see in model 1.

Going forward, we could try to see how model 2 would perform with the scaled data set used with model 1. It uses some different collaborative filtering methods than model 1 and may prove to be better or worse than its current configuration. More users could also be included in the training to raise precision of the recommendations closer to the likes of model 1. When scaling the data in model 1, we could have also spent more time attempting to remove outliers such as those individuals who only play one game for a significant amount of time. That seems to have affected the scaling of the ratings of the rest of those players' games greatly. Model 2 dealt with these types of biases in better ways than model 1. Another idea is to combine content-based filtering techniques with our collaborative filtering approaches. If we had more data about the games played like their genre, story type, art style, and leveling system, we may be able to better determine user sentiment, as well as measure how similar a player's played games are to the ones our recommendations systems are recommending to him or her. Lastly, more data about the user would have been helpful. We only had access to their hours. If we had been in possession of their actual reviews, wish-listed items, and demographic data, we may have been in a better place to offer suggestions for the type of titles an individual may want. Overall, the project proved challenging, but very beneficial in giving us real insight into how AI can be used in a real world application. It set a strong foundation of interest for future projects and taught us the importance of not only having good data, but the ways in which you can extract valuable meaning from data.

VII. References

[1] A. Biswas, K. S. Vineeth, A. Jain and Mohana, "Development of Product Recommendation Engine By Collaborative Filtering and Association Rule Mining Using Machine Learning Algorithms", 2020 Fourth International Conference on Inventive Systems and Control (ICISC), Coimbatore, India, January 2020, pp. 272-277, doi: 10.1109/ICISC47916.2020.9171210.

[2] S. J. Russell and P. Norvig, Artificial Intelligence: A Modern Approach, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[3] M. H. Mohamed, M. H. Khafagy, and M. H. Ibrahim, "Recommender Systems Challenges and Solutions Survey", 2019 International Conference on Innovative Trends in Computer Engineering (ITCE), Innovative Trends in Computer Engineering (ITCE), 2019 International Conference, Aswan, Egypt, February 2019, pp. 149–155, doi: 10.1109/ITCE.2019.8646645.

[4] D. J. Dudhia, S. R. Dave, and S. Yagnik, "Self Attentive Product Recommender – A Hybrid Approach with Machine Learning and Neural Network", 2020 International Conference for Emerging Technology (INCET) Emerging Technology (INCET), 2020 International Conference, Belgaum, India, June 2020, pp. 1–4, doi: 10.1109/INCET49848.2020.9154034.

[5] F. R. Rehman. "Min Max Normalization in data mining." T4Tutorials. https://t4tutorials.com/min-max-normalization-of-data-in-data-mining/ (accessed December 15, 2020).

[6] S. Marafi. "Collaborative Filtering with Python." Salem Marafi. http://www.salemmarafi.com/code/collaborative-filtering-with-python/#:~:text=Collaborative%20Filtering%20with%20Python%201%20Refresher%3A%20The%20Last.FM,collaborative%20Filtering.%20...%204%20Entire%20Code%205%20Referenence (accessed December 15, 2020).

[7] Santhosh. "Recommendation Systems: Collaborative Filtering just with numpy and pandas, A-Z." Medium. https://medium.com/@sam.mail2me/recommendation-systems-collaborative-filtering-just-with-numpy-and-pandas-a-z-fa9868a95da2 (accessed December 15, 2020).

[8] Kaggle User, "steam_game_recommender." Kaggle. https://www.kaggle.com/jwyang91/steam-game-recommender (accessed December 15, 2020).

[9] Tamber, "Steam Video Games Version 3." (2016). Distributed by Steam. https://www.kaggle.com/tamber/steam-video-games (accessed December 15, 2020).

[10] PACKT Publishing, Birmingham, England. Building recommendation systems with Python (2019). Accessed: December 15, 2020. [Online] Available: http://search.ebscohost.com.umasslowell.idm.oclc.org/login.aspx?direct=true&db=edsasp&AN=edsasp.ASP4740624.marc&site=eds-live

[11] Wikipedia. "Collaborative filtering." Wikipedia. https://en.wikipedia.org/wiki/Collaborative_filtering (accessed December 15, 2020)