ISyE 6740 – Spring 2021
Final Paper

Team Members: Benson Igarabuza 903566066, Samantha Virgil 903651249

Project Title: Video Game Recommendation System for Steam Users

Introduction/Problem Statement

As a customer, viewer, gamer, or any user that can benefit from suggested products or services, the quality of suggestions and recommendations can greatly improve your experience. When purchasing products, the only data available is the purchase data, not how often you use and enjoy the product later on. A customer can rate a product or service based on their satisfaction, but many customers don't bother with ratings. The customers that do rate items, tend to only rate if they had an extremely positive or an extremely negative experience, resulting in bias in the overall ratings for various products and services. Instead of the typical star classification rating scale, some companies, like Netflix, offer a 'like' or 'dislike' feature so it's faster and more convenient for the user and therefore more likely for the user to rate what they like. This ultimately leads to better recommendations. For streaming services and video games, other data is collected besides the typical rating that can be used to indicate a user's satisfaction with a product or service. Steam is an online, cloud-based gaming library where users can purchase and play video games. Steam collects user data on things like play time per game, how many achievements a user has unlocked for each game, the genre of games, multiplayer capabilities of games, etc. So, how can we use this unique data to better understand the user and make more accurate recommendations when a user doesn't rate what they like? We are interested in finding the solution to this and seeing how we can recommend games to Steam users based on their data in lieu of ratings.

Data Source

The data used for this project is part of the paper - *Condensed Steam: Distilling the Diversity of Gamer Behavior* published by O'Neill et al [1] and presented at the 2016 Internet Measurement Conference. The raw dataset is ~160GB and includes various attributes that the authors collected using the Steam API and the Steam desktop client. The API was used to source user-based attributes while the desktop client was used to obtain game attributes. The dataset is saved as a MySQL back up file which we loaded to a local instance for our analysis. While the dataset was exhaustive, we focused on the below tables from the dataset:

Raw Data Attributes:

Table	Fields
APP_ID_INFO	appid, Title, Type, Price, Release_Date, Rating, Required_Age, Is_Multiplayer
GAMES_DAILY	steamid, appid, playtime_forever, dateretrieved
GAMES_GENRES	appid, Genre

The APP_ID_INFO table held game-based attributes. The attributes we used for our model were 'appid', 'Genre', & 'Price' as these were the most relevant and complete attributes in creating our recommendation engine. We had initially hoped to use the 'Rating' attribute however the field did not represent ratings made by users in the dataset. The 'Rating' attribute here represents the game rating on Metacritic. Rather than relying on this rating, we derived a rating based on user game play time.

The GAMES_DAILY table held records of user game play time at different daily retrieval times from the Steam API from October 10, 2014, through November 9, 2014. This served as snapshots of user-game data state over time. Oddly, the number of records on each date varied wildly. To get a robust dataset for our model, we elected to use the November 9th, 2014, snapshot as it had one of the highest number of records. In addition to the fields outlined above in the GAMES_DAILY table, we computed a new field named: `pseudo_rating'. This field was computed by taking each user's playtime_forever for a particular game divided by the user's total playtime_forever across all games.

The GAME_GENRES table held records of what genre each game belonged to. We found that each game in the dataset did not necessarily belong to one game genre. In fact, some games belonged to seven genres. This presented to be an obstacle to our Nearest Neighbor model as our model assumes that each game only has one genre. We therefore randomly assigned one of the genres for instances where a game had multiple genres.

Fields used to feed our model:

Field	Description
appid	Unique identifier of game
title	Game title
genre	Game Genre
price	Price of Game
steamid	User id
pseudo_rating	Game rating based on user's playtime relative to the user's aggregate game playtime across all games

The authors have made the dataset publicly available at: https://steam.internet.byu.edu/. Other tables in the dataset include ACHIEVEMENT_PERCENTAGES, APP_ID_INFO, FRIENDS, GAMES_DAILY, GAMES_DEVELOPERS, GAMES_PUBLISHERS, GROUPS, & PLAYER_SUMMARIES. These attributes were interesting to see in the dataset, but we believe that a recommendation system based on player play time would be a good proxy for true user game ratings. This would be better than the Metacritic rating included in the APP_ID_INFO table as actual user behavior is a good indicator of how the game is liked. This approach of using this pseudo rating in conjunction with our Nearest Neighbor model is discussed further below.

Methodology

To recommend anything to a user, it's important to gather information and use it to understand the user's behavior. For video game players, or 'gamers', we can understand more about their behavior by observing how they split up their play time between different games. In the data we gathered and as suspected, many users don't have recorded ratings for the games they've played, so we needed to create a metric that represents how much a player likes each game they've played. We concluded that the amount of time a user commits to a single game accurately measures how much they enjoy that game. A user won't commit days of play time to a game that they don't enjoy. A player will also tend to spend the most time playing the games they like the most.

We chose to create two metrics based on users' playtime. We calculated a 'pseudo rating,' mentioned above, for each game a user has played as a proportion of play time for each individual game divided by the user's total play time over all games played. We then calculated an 'average rating' for each game by taking the average 'pseudo rating' for each game across all players.

To recommend games a user is mostly likely to enjoy, we concluded it would be best to make recommendations based on each user's most played game. To find and store a user's highest played game, we filtered the data by user, then by highest pseudo rating, and selected the appid of the highest pseudo rated game on the user's list. We then went to our APP_ID_INFO table and selected the game with the same appid we selected from the user's data table. From the APP_ID_INFO table, we extracted all attributes of the game that we want to use to find similar games with. We decided that 'genre', 'average rating' (calculated using users' pseudo ratings), and price, are the features that best characterize each game.

With the extracted game data for each user's top played game, we were able to create a model for finding the games that are most similar to each of those user's top played games. We chose to use the Nearest Neighbors function from Python's Scikit-Learn library to find the games closest to the user's chosen top game. We used the APP_ID_INFO data as the training data and the user's chosen game as the testing data point. We used Manhattan distance for our distance function as we concluded it would be best for our data. We had the model return the 10 closest games to the user's chosen game. We then filtered those games by removing any games that were currently present in the user's previously played games list. To do this, we extracted a previously played games list for each user by filtering the user data by user and extracting the game title column. We then sorted the game titles in the recommended list of games alphabetically so that when different editions of the same game were recommended, they would show together in the list and in order of release date. During the filtering stage, we also included a count for how many games were removed/filtered out of each user's list to use it as a potential evaluation method.

Evaluation and Final Results

To evaluate the output of our recommendation engine, we needed a way to compare the list of recommended games to the user's chosen game. We originally counted how many games were filtered out of the recommendation list. We figured that if the engine is removing multiple games from the original recommended list, then the engine is choosing games the user is likely to enjoy, since we know the user has already played those games. After analyzing the output, we found that the engine mostly only removed one to two games, one of which always being the user's top game. We then rendered that method of evaluation to be unsatisfactory. The model that we chose is an unsupervised machine learning method, which means there is no real feedback mechanism. We decided to randomly select user IDs and manually analyze the engine's output. We found that many users had amazing recommendations, but others not so much. For example, below is the evaluation of three random users.

User 1 Top Game: Far Cry 3 Recommendation List: [Borderlands, Call of Duty: World at War, Call of Duty® 2, Call of Duty® 4: Modern Warfare®, Call of Duty®: Modern Warfare® 2, GUN™, Killing Floor, Mount & Blade: Warband, Street Fighter® IV]	User 2 Top Game: Call of Duty®: Black Ops Recommendation List: [Borderlands: The Pre-Sequel, Call of Duty®: Modern Warfare® 3, Call of Duty®: Modern Warfare® 3, DARK SOULS™ II, Deadpool, Prototype 2, Shadow Warrior, The Amazing Spider-Man 2™]	User 3 Top Game: Warhammer® 40,000™: Dawn of War® II Recommendation List: [Age of Empires II HD, Command & Conquer 3: Kane's Wrath, Dominions 3: The Awakening, Majesty 2 Collection, Medieval II: Total War™, Napoleon: Total War™, Panzer Corps, Sid Meier's Civilization® IV, Warhammer® 40,000: Dawn of War® II Chaos Rising]

For User 1, ‘Far Cry’ is a first-person shooter game. ‘Borderland’s, all of the Call of Duty games, and ‘Killing Floor’ are all first-person shooter games. It is very likely that someone that enjoys ‘Far Cry’ would enjoy any of those games listed. The games ‘Mount & Blade’ and ‘Street Fighter IV’ don’t quite make sense though. ‘Mount & Blade,’ though still a fighting game, is more of a strategic war game, rather than a first-person shooter or something of the like, and ‘Street Fighter IV’ is a fighting game, which is not similar at all to ‘Far Cry.’ It’s possible someone spending a lot of time playing ‘Far Cry’ would not be interested in ‘Street Fighter IV’.

For User 2, ‘Call of Duty: Black Ops,’ is again, another first-person shooter game. Our engine recommends ‘Borderlands: The Pre-Sequel,’ a few Call of Duty games this user has not played yet, and the remaining games don’t seem quite accurate. The remaining games are action-adventure games. Some fit a similar theme to ‘Call of Duty: Black Ops,’ however, the content of the others is pretty far off. ‘Deadpool’ and the ‘Amazing Spider-Man 2’ games probably are not very good recommendations for this user based on their top game, but they were most likely recommended based on price. We were also surprised to see that other Borderlands games were not recommended, only the prequel was.

For User 3, Warhammer 40,000: Dawn of War II is a real-time strategy-tactical role-playing video game. All of the games our engine recommended are real-time strategy games that this user is highly likely to enjoy. The recommendations made for user 3 I believe are the best out of the three, with the recommendations for user 1 being good and the recommendations for user 2 being fair. No user of all of our users (not just these three) had poor recommendations, so we’ve concluded our recommendation engine has done a great job. There are a few reasons for some recommendations that didn’t seem very close to some of the users’ top games. It could be that the user’s top game was a low-cost game, but many similar games are actually high cost, so they were considered less ‘similar’ in our model and vice-versa. It could also be the case that the user chose a low overall rated game (using the ‘average rating’ discussed above), so that game was considered more ‘similar’ to games of different genres. By the nature of video game genres and the fact that genre is a categorical variable, there is no way to sort genres by any metric. When considering this in the Nearest Neighbors algorithm, games could be close to other games that are not similar in regard to genre at all. We also need to consider that for games with multiple genres, we selected one at random, so some games could be listed under a genre that does not describe it the best.

Conclusion

Overall, our model performed very well. Many of the recommendations made by our engine were relevant compared to each user’s top played game. To improve how relevant our engine’s recommendations are, we could take a user’s top three played games rather than only the most played game and return recommendations based on all three. This way if this user enjoys more than just first-person shooter games, we can recommend them a wider variety of games they might like, making the recommended games more relevant to the user. To improve our games filter for the user’s final recommended list of games, we could implement a function to remove multiple games from the same franchise. For example, User 1, who’s recommendations were evaluated above, was recommended 4 different Call of Duty games. The user will probably get the hint they should check out some of the available Call of Duty games after the first one or two Call of Duty games in the list; they don’t need to see 4. We could also consider a better way of handling games with multiple genres. Assigning one genre randomly has most likely affected the model’s performance. Genre is also one of the most important variables when it comes to the nature of video games and our model. We could go further and find a way to assign weights to the variables to make sure the genre variable has a higher significance, or weight, in the model.

Work Assignment

In terms of work/tasks assignments, Samantha Virgil came up with the idea and was primarily responsible for designing the model approach, loading the dataset to the model & evaluating the results. Benson Igarabuza was primarily responsible for the data retrieval, loading the datasets in a local MySQL instance, setting up the relevant SQL views, and setting up the coding infrastructure of the project in Github (https://github.gatech.edu/bigarabuza3/ISYE6740Project). All team members have contributed a similar amount of effort.

References

[1] Mark O'Neill, Elham Vaziripour, Justin Wu, and Daniel Zappala. 2016. Condensing Steam: Distilling the Diversity of Gamer Behavior. In Proceedings of the 2016 Internet Measurement Conference (IMC '16). Association for Computing Machinery, New York, NY, USA, 81–95. DOI: https://doi.org/10.1145/2987443.2987489