Collaborative Filtering and Innovation: A Case Study from Steam Recommendations

Coen D. Needell

Tue, Mar 16, 2021

Introduction

The modern world is flooded with options; which TV show to watch, which video games to play, which espresso machine to buy. E-commerce is characterized by its quantity. This doesn't, however, imply that e-commerce platforms are inundated with low quality goods. Truly fantastic media is released every year. 2019 saw the release of a game *Disco Elysium* which has been lauded as one of the best CRPGs (Computer Role Playing Games) of all time, even though the so-called golden age of the genre is long past(Metacritic, n.d.). 2019 also saw the release of 8033 games on Steam, PC gaming's most used distribution platform (Statista, n.d.). Television has experienced a similar trend, with some calling this the Golden Age of Television Writing. However, these trends present a problem. Clearly not all 8033 games that were released onto Steam in 2019 were brilliant, genre-redefining masterworks like *Disco Elysium*, *Outer Wilds*, and *Untitled Goose Game*. Players don't have the money and energy to invest in games that they aren't going to enjoy. They need someone to separate the wheat from the chaff for them. The natural inclination is to find some way to automate this process, that way people can get recommendations that are tailored to them, and it can be done with minimal human labor. The problem is, these automated systems can have unintended consequences, like stifling innovation.

Netflix famously tried to address this problem with a million-dollar competition to design a new content discovery algorithm. While the current algorithm is proprietary, from prospectus documents, the Netflix prize algorithms, and press releases, it can be surmised that Netflix's recommendations come from a mixture of clustering and collaborative filtering, incorporating both user data and content data(Portugal, Alencar, and Cowan 2018). Netflix lives and dies by its recommendation algorithm, so it's become fairly good over the years. Steam, however, is in a very different situation. While Steam puts a lot of resources into making better content discovery tools, it's less focused. Steam has, by my count, eight different content discovery systems. The "discovery queue," the "interactive recommender," the "play next shelf," the "deep dive" (deprecated), the "community recommends" page, the "sale explorer," the "navigator," and the front page has an infinite scroll-style recommendation system. All of these employ some aspect of personalization or machine learning inference.

Steam is, by it's very nature, overwhelming to the consumer. This is a side-effect of the fact that there's a very low barrier to entry to get on Steam. Your product just needs to not break any laws, be your right to sell, not threaten the security of computers it's installed on, and not be pornography to get in[1](Steamworks, n.d.). It's part of Steam's core philosophy that anyone using their platform succeeds or fails under their own power, rather than because Steam themselves decided that their game should succeed[2]. For contrast, Steam's biggest competitor, the Epic Games Store, for example, has highly restrictive barriers to entry, but also guarantees sales (Epic buys the first X copies and resells them, taking the loss themselves if they don't sell) and promotes new releases heavily. Steam leans into the quantity approach, but it also has plenty of tools to help a player sift through the thousands of games. In addition to the AI driven techniques above, there are also human driven systems, like a curator system, where anyone can create a curated collection of Steam games and people who find they have similar tastes can get recommendations that way, this also keys into the "discovery queue"(Kuchera 2017). Every game on steam has an open review space, where anyone who has bought and played the game can leave a review with a binary recommendation tag. These systems are known to have their problems as well, such as instances where a game is inundated with mobs of bad reviews in retaliation to something unrelated the developer did. These concerns are

[1]This bar is even lower than you think, as we will discover.

[2]Valve does sell some of their own products on Steam, but they don't really push them very hard. I can't remember the last time I saw any Valve games come up in my Steam recommendations.

outside the scope of this discussion, however, we are going to look at the problems with automated recommendation.

Statistical methods for recommendation are a great way to get information on large groups of people. They're even quite good at grouping people together. They also make fantastically accurate predictions in well-trodden ground, but when you go off the beaten path, their accuracy falters. Statistical prediction systems of any kind are playing a game of averages and expectations. More sophisticated methods perform better on new situations because they're better at recognizing patterns, but they still can be tripped up when presented with patterns they don't know how to parse. Ultimately, they use the patterns they do recognize, and make a prediction based on the average of things they've seen exhibit similar patterns. So here's the problem, game developers, and the media industry in general, are in the business of doing things that have never been done. It is, in a sense, their jobs to go off the beaten path. Most recommendation algorithms, and indeed the technique that this discussion looks at, are based on the assumption that if two users liked the same things in the past, then they'll like the same things in the future. This is not a bad assumption in general, but there are cases where it fails. For a thought experiment, consider two groups of gamers, the Strategy Buffs and the Casuals. Casuals generally prefer clever gameplay, engaging, but not overwhelmingly deep. They tend to gravitate towards simulation and puzzle games. The Strategy Buffs, however, love deep tactics, they like the feeling of outsmarting an opponent. There is some overlap in the games these groups enjoy, but not a lot. Now lets say a game like *Wargroove* is released. It's a tactics game, but it doesn't run too deep, it's about outsmarting your opponent, but it doesn't have long, grueling campaigns, it can be played in thirty-minute chunks while watching TV or between getting home from work and cooking dinner. It appeals to some Strategy Buffs because it has some innovations in the Medieval-Tactics subgenre, and it appeals to Casuals because it's easy to pick up, and it has cute graphics. Now imagine a user who plays *Wargroove*, and likes it, and wants to use the recommendation system to find more games like it. What is the optimal result? Even further, imagine we have this game that has been played and enjoyed by some people in the Strategy Buffs group, and some people in the Casual group. Recommendation systems based on clustering users would consider it to be a poor match for members of either groups. These are two of many quirks of the recommendation problem, and they're ones that can stifle innovation if not properly handled.

The Recommendation Problem

From the point of view of the consumer, there are N options of varying quality. Each of them is unique. In addition to their quality, there's also a sense of how good of a match that option is for the consumer. Some people like action, some people like to play online, some people prefer strategic thinking. In addition to people's general preferences, they also have in-the-moment preferences. This is often described as context-sensitivity. It can be as simple as, later in the day you may be more in the mood for something more intensive, a multi-hour experience; early on Saturday morning you may prefer something lighter. We also have to consider the fact that this is an incomplete-information problem. A new user usually signs up for a media service for one thing in particular. For Netflix it may be to watch *Avatar: The Last Airbender*; Steam exhibited its initial growth in user count when *Team Fortress 2* became Free-To-Play[3]. On the other side, we have access to a large group of consumers, and a large group of options. We can see user behavior, and we can see what content they're consuming, and some proxy for their enjoyment of it. The user behavior data is limited. Valve can only see a player's playtime on games they own on Steam. They can't see, for instance, how much time a player spends adventuring in *Pathfinder: Kingmaker* as opposed to managing their castle. These are two different play-styles within the same game that could yield information about the player, but all Steam sees is the combined time playing the game. Steam also cannot see what a player plays on other platforms. Say a player is a huge fan of Racing games, but happens to play them all on Xbox or only buys them from the Windows Store[4]. Steam will never know to recommend Racing games to that player, even with some incredibly sophisticated algorithm.

[3]This is a monetization model where most players of the game play for free, and profit is made by selling in-game items to the hardcore players. It is not without its controversy, although *Team Fortress 2* is often held up as a positive example of a Free-To-Play model.

[4]Not as absurd as it seems, many racing games in the last couple of years have been released as so-called "Microsoft Exclusives," such as *Forza Horizon 4*.

As mentioned before, usually services try to solve this problem with Collaborative Filtering. A class of algorithms that compare a user's tastes to other user's tastes to find similarities, and formulate a recommendation based on that. Generally, the algorithms go like this:

```
given a user, u
create a vector of u's known tastes
for u_prime in users:
    create a vector of u_prime's tastes
    calculate the similarity between u and u_prime
    multiply u_prime by the similarity
sum all of the weighted tastes along each item to create a vector of scores for each item
recommend the highest scoring item
```

It seems like a pretty good algorithm. Weight people's tastes by closeness to your taste, then combine them and see what "people who like the things you like" like. Here's the problem, how do all of the u' find out what they like? Well, going by this algorithm, they would try out the games that people like them like. At some point, for new games to get into the recommendation ecosystem, someone needs to be seeking them out, outside the recommendation ecosystem. Imagine a new game with a less-traditional target audience (like the aforementioned *Wargroove*) is introduced to the ecosystem. Some players seek it out, or were involved in some outside community that's talking about it, or some other mechanism occurs with the end product being that a small subset of the target audience plays the game on Steam. Now, a user who doesn't know about the game, but is in the target audience logs onto Steam. The collaborative filtering algorithm runs. The thousands of people playing *Sid Meyer's Civilisation VI* and *Europa Universalis IV* outweigh the hundreds of people playing *Sid Meyer's Civilisation V* and *Wargroove*, so the player is recommended *Europa Universalis IV*. There's yet another layer to this feedback loop; over time, as the player who would originally fit into the Casual/Strategy group keeps getting recommended the hardcore strategy games like *Europa Universalis IV*, they eventually try one, and get into it, and their tastes may shift by virtue of the recommendations always nudging them in the more populous direction(Ransbotham, n.d.).

Consider also, the fact that these algorithms rely on a small subset of the population to seek out content without aid of the recommendation algorithm. Where are they going? For video games, these recommendations are coming from trade magazines, online forums, and press releases from developers that the player is already a fan of. While some trade magazines (Rock, Paper, Shotgun; Indie Gamer) try to focus on smaller developers, the majority of them will write reviews for well-known projects. Even among the independent-focused magazines, it's fairly uncommon that they will dig up some completely unknown project and bring it to the light. This phenomenon is even more exacerbated in the case of freshman studios, especially those headed by people new to the industry. As for online forums, you see some projects gain popularity this way. There's a very low barrier to entry, but many more posts fizzle out than go viral. Developer press releases occupy a strange intermediary space, where there's very low barrier to entry, but will also be only viewed by people who have been interested in that developer's past work, or are already planning to buy their next game, so this source of news does very little for freshman studios.

The conclusion to draw from all of this is that recommendation systems that use collaborative filtering will only exacerbate any problems that arise from other forms of content discovery. Theoretically, a few high-similarity users can be drowned out by medium-similarity masses; big, established studios get more recommendations because they have more people discovering their games from other sources, and small studios with niche audiences get pushed down the list, even if they may be a better fit. This, in conjunction with the fact that the recommendation engine influences tastes and not just behavior, creates a feedback loop in which innovative developers have a harder time succeeding in the marketplace.

Methodology

Data

To illustrate how collaborative filtering unfairly weights more popular games, we will look at two datasets. The first is a dataset of all user's playtime who have set their games list to public(Tamber 2017). This is a small subset of the whole Steam user-base, and probably is not a representative sample, so for the purposes of this discussion, "popularity" refers to popularity in the sample, as to control for any correlation between tastes and presence in the sample. The data relates anonymized user ids to games that the user has purchased, and games that the user has played. For "play" events, the dataset also reports the user's total playtime. The dataset doesn't report the steam official `appid`, nor the Steam official game name, but it reports a "common name" for each game. For example, *SPORE* is reported in the data as *Spore.* This database records 200,000 user events across 11,350 users and 3337 games.

The second dataset was constructed using the "SteamSpy" API(Galyonkin 2021). SteamSpy is a third party service that records and makes public a database of all of Steam's official information endpoints, recorded daily. SteamSpy was used instead of the Steam official API because of its ease of use, and because it deliberately does not record information on games that do not sell at all. This database was gathered over the course of a couple days in late February, 2021, and contains 26,862 games, each with a complete description of "tags." Steam's tags are user-submitted suggestions for categories to place each game in, and this database has a list of tags associated with each game, as well as the number of users that voted in favor of that tag, we also have a rough estimate of how many people own the game on Steam. It also has the Steam official `appid` and the Steam official game name.

First, we have to deal with the problem that these are imperfectly related datasets. For this I'm using the `python` package `fuzzywuzzy`("Seatgeek/Fuzzywuzzy" 2021). The package uses Levenshtein Distance to calculate the ratio of similarity between two strings. In finding the `appid` for each common game name, I am making the assumption that the official game name is the Levenshtein-closest name in the larger database. This works fairly well, although there are some notable exceptions. For example, *Dead Island: Epidemic* was an experimental MoBA[5] title, based off of the popular First-Person-Shooter *Dead Island. Dead Island: Epidemic* is no longer available on Steam, as it was cancelled during an open beta-testing period. Some of our user sample played it, and it's Levenshtein-closest game in the SteamSpy dataset is *Dead Island: Riptide*, a standalone expansion to the original. That aside, this method is only inaccurate in very specific cases.

Next, we construct a sparse matrix, where rows represent users, and columns represent games. For each user u, the matrix has a vector $r_{u,i}$ that represents the user u's playtime in game i. We normalize all of these vectors to using an L1 norm. Any new user will have their playtimes used to construct a new vector such that all of the indices are in the same order as the columns of this matrix. If a user has played a game that doesn't exist in the matrix, a new column is added to represent that game.

[5]Technically stands for "Multi-player Online Battle Arena," but that can describe pretty much any competitive online game. Generally "MOBA" describes games from a top-down perspective, where you control one character, presented as a commander in a war-zone. Ur-Examples include games like Defense of the Ancients (DotA) and League of Legends (LoL). This genre is also defined by heavy use of acronyms.

Content Analysis

On the other side of things, we take all of the games known to SteamSpy, and we construct a similar matrix $v_{i,t}$. This is along the tag votes. It is also normalized along the rows using an L1 norm. These vectors were then fed into a k-means clustering algorithm("Scikit-Learn/Scikit-Learn" 2021). Using the "elbow" heuristic, considering both the inertia of each cluster's centroid and the average distance between each game and its nearest centroid to optimize for k, we end up with fourteen clusters that can be qualitatively described as:

Cluster	Label
0	Action Simulation
1	Japanese-style Role-Playing and Adventure Games
2	Casual Puzzle-Adventures
3	Indie games with dark themes
4	Free-To-Play Shovelware[6]
5	Violent Shovelware
6	Logic/Classical Casual
7	Casual
8	Strategy Simulation
9	Adventure Shovelware
10	Tactics
11	Software
12	Action PvP
13	Skill-Based Adventure

The qualitative descriptions come from looking at the games closest to the centroid, as well as largest tag-dimensions of each centroid. We end up with some weird categories. As we already discussed, there are a lot of low-quality, and borderline terms-of-service breaking games. While this clustering method does give us some interesting categories to look at, the categories which contain the most popular games on Steam are too broadly defined, and the categories which contain all of the shovelware are too narrow. For example, open-world RPGs are by a wide margin the most popular genre of single-player games, and they don't fit cleanly into any of these categories. So let's try that k-means clustering again, but we'll weight each game by SteamSpy's ownership estimates. Again, using the elbow method, we end up with 7 categories.

Cluster	Qualitative Label	Number of Games	Number of Users in Dataset
0	Multiplayer Action Non-FPS	276	1870
1	FPS Multiplayer	285	17791
2	Adventure/RPG	1590	24954
3	Tactics/Strategy	543	7301
4	Casual	280	1565
5	Indie/Anime Story-Driven	91	368
6	Action Free-to-Play	272	16640

These much more cleanly match the experience of the major game genres. Category 5 is the only one that seems weird, but it makes a little more sense when you consider how many games on Steam are Visual Novels or otherwise text-driven games. This includes some extremely successful games like *Pyre*, *Doki Doki Literature Club*[7], and others. Even though there are thousands of games on Steam that fit into this category, they aren't generally popular. Since our dataset is a random sample of users, there aren't a lot of these games in our sample. We also see the Multi-player market dominate the diversity of these clusters. This is not super surprising. The top-played games on Steam are consistently Multi-player games. We've also recorded how many games are present in each cluster and our user-playtime database.

[6]"Shovelware" generally describes games, or more generally software, that are created cheaply, using recycled or sometimes pirated assets. They often cross into the pornographic, and are usually intended to make profit quickly with low cost.

[7]It's a horror game.

Recommendation Engine

We'll use a collaborative filtering system based on user playtime, using $\cos(\vec{x}, \vec{y})$ as a measurement of similarity. This will be fairly robust, since we've normalized each vector using an L1 norm. That way, someone who has played *Sid Meyer's Civilization V* for twelve-thousand hours and *Anno 2070* for ten hours will be roughly identical to a player who has played *Sid Meyer's Civilization V* for ten hours and nothing else. Following the algorithm from before, for each user u over games i, we estimate a rating $r_{u,i}$ as:

$$r_{u,i} = k \sum_{u' \in U} \text{sim}(u, u') r_{u',i}$$

Where

$$\text{sim}(u, u') = \frac{\sum_i r_{u,i} r_{u',i}}{\sqrt{\sum_i r_{u,i}^2}\sqrt{\sum_i r_{u,i}^2}}$$

Which is the same as the cosine between vectors $\vec{r}_u$ and $\vec{r}_{u'}$. Then, we recommend to the user u the games with the highest predicted rating, skipping over ones they already own.

In order to probe the recommendation engine, we create a system that randomly draws games and generates playtimes. The games are drawn from each cluster independently, without replacement, and to generate playtimes, we sample from the space of known playtimes. That way we get a distribution of playtimes which we know is realistic. With these tools we can feed in users with different profiles and examine the recommendations. We'll compare these recommendations to a content-based approach. By using our tag database, we'll put in a set of games, normalize and weight their tags by the player's playtime, and then recommend a game based on the criterion $\frac{cos(u,i)}{2} + \frac{r_i}{2}$, where r_i is the lower bound of the confidence interval created using binary user reviews to estimate quality.

Results

First up, a player of niche games from a populous category. Drawing five games from cluster 2 gives us:

- *The Basement Collection*
- *TypeRider*
- *Dynasty Warriors 8 - Empires*
- *Port of Call*
- *Wind of Luck Arena*

The algorithm recommends *Counter-Strike: Global Offensive.* An extremely popular PvP action game. This is very clearly exhibiting the behaviors we hypothesised. The content-based approach recommends *MagiBot*, a little skill-based game with strategy elements. It has 8 user reviews, all positive, although they criticise its short length and general lack of difficulty. Seeing as how this randomly drawn player enjoyed *The Basement Collection*, a series of small, difficult skill games, and *Type: Rider*, a short, relaxing adventure through the history of typefaces, as well as *Dynasty Warriors*, a series of strategy games, they probably would benefit from trying *MagiBot*, based on my own intuition.

Now let's put in some popular games. Putting in a player who has played 3 mainstream RPGs for equal amounts of time:

- *The Elder Scrolls V: Skyrim*
- *Borderlands 2*
- *The Witcher 3: Wild Hunt*

The algorithm recommends *DotA 2*, again, an extremely popular PvP game. Even when it should be an easy answer, the algorithm prefers a more popular game. The content-based approach recommends *The Elder Scrolls IV: Oblivion.* A great, if a bit outdated, recommendation for someone who enjoys mainstream RPGs.

Now let's try a user who doesn't fit cleanly into one of the content clusters. We'll draw two games from category 5, the story driven games, and two from category 3, the tactics/strategy games.

- *Cities in Motion 2*
- *Steam Marines*

- *500 Years Act 1*
- *Lili Child of Geos*

The top recommendation is *Terraria*, which doesn't really fit into any of these categories. It is, however, one of the most popular games on Steam. The content-based approach isn't amazing either, it recommends 纸境英雄 *Papercraft*, an experimental indie game with a couple-hundred reviews, 84% of them positive. It pretty firmly falls into category 3, although it shares a visual similarity to category 5, and does have the "Indie" tag as a major tag on Steam.

Now that we have a general idea of how this algorithm performs, let's probe it more generally. We will generate a profile by drawing five random games from each cluster one thousand times. Then we can create a matrix of how many times out of one-thousand a game from each cluster is recommended. If the hypothesis is correct, and the recommendation engine leans toward more popular categories, we should see the more popular clusters 1, 2, and 6 get recommended more often than the others. The results are shown graphically in Figure 1.

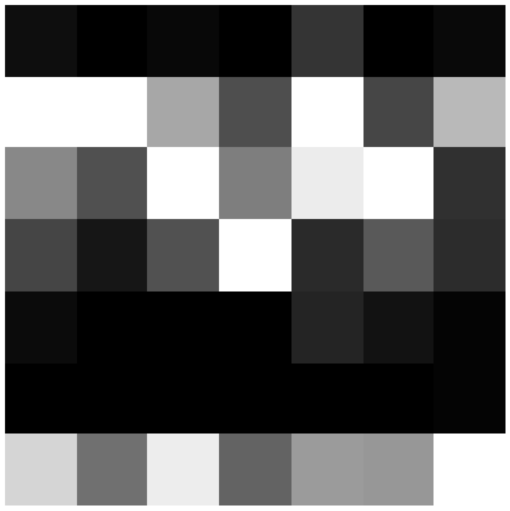

Figure 1: A heat map showing the correlation between input cluster and output cluster. Inputs are along the x axis from left to right, and outputs are along the y axis from top to bottom. White indicates a large number, black indicates a small number. Each column has been standardized.

And while we do see some clustering around the diagonal, most of the recommendations fall into categories 2 (Online Competitive games, mostly First Person Shooters), 3 (Adventure RPGs), and 6 (Free-to-Play online games), no matter what cluster we choose as input. Cluster 3 seems to be exempt from this trend, but Strategy games take up the middle in popularity. This supports the hypothesis collaborative filtering is more influenced by popularity than by the individual's tastes.

Discussion

Now, let's consider the implications of this. If you're logging on to Steam and the majority of your recommendations are for the most popular games, no matter what games you're playing, then what good is the recommendation engine doing you? What good is the recommendation engine doing the community? As discussed earlier, Steam presents itself as a platform where quality floats to the top. While it's true that a lot of the games that appear in steam recommendations, on it's front page, and through it's other tools are often of quality, this set is not necessarily complete. From the developer's standpoint, quality doesn't translate to success in the way that Steam promises. Take for example that game we dug up with our simple content-based algorithm, 纸境英雄 *Papercraft*. I might actually go try that out after I'm done writing. It's stylish, and from the reviews it looks like people enjoy it. It seems to have the quality, but not the following. Sometimes it takes years for games to gain the following. Take for example *Among Us*, it's a fun little implementation of the children's game Werewolf or Mafia, it was released quietly onto Steam three years ago to little fanfare, and then exploded to be highly popular in 2020. Compare this to *Forza Horizon 4*, a port from the Windows Store. It released onto Steam a couple of days ago at the time of writing. According to SteamSpy, it has already sold around 100,000 copies. Yes, it's a quality game, but it also had a following before being released.

Given that recommendation engines favor the externally popular, then independent studios can't rely on discovery tools to help them gain attention. Since independent studios often don't have the cash on hand to advertise, don't have the connections to get reviewers to review their games, and don't have big publishers to help them along, then they're forced to rely on luck, and forum chatter. Word-of-mouth helps a lot, but it's often slow, and unless they get lucky even the best freshman games may not see profits until a few months after release.

In addition, we've seen that when a developer releases games in less popular categories, they're less likely to recommended to people, even people who historically only play games in those categories. This means that, in terms of recommendations, developers are rewarded for making safe games with broad appeal. While this is true in media generally, it's usually the role of independent media studios to push the proverbial envelope, and when successful, they're rewarded by their niche audience. What we've seen here is that even if you do succeed in making a good, innovative game, it won't even be recommended to the kind of people that would like it. Those people only get recommended *Grand Theft Auto 5* and weird soft-core-pornographic puzzle games.[8]

If, as some authors suggest, recommendations shape our tastes in a feedback loop, then we're doing more than just punishing developers for innovating, it could be making the entire environment less conducive to innovation. It's probably not the case that these effects are radical, I will probably never become a hardcore competitive action game player, but they could be subtle, like pushing lovers of casual simulation games like *Stardew Valley* into the more popular Adventure/RPG space.

More broadly speaking, even though the case has been video games on Steam, it's clear that the argument works in other places where collaborative filtering techniques are used. Because of the collaborative nature of the algorithm, the results are always going to be skewed towards the tastes of the masses with small changes in the direction of the individual. While this doesn't really make it harder for people with niche tastes to find more of what they enjoy, since they tend to be used to the notion of searching outside the platform for recommendations, it does make it harder for studios that want to make things for that niche audience to be found. In the end, if we want these recommendation systems to work for everyone, they need to incorporate more than just user data, they need to tap into content somehow. It's a much harder problem to solve, but it would decrease the barrier-to-entry for both studios and consumers.

[8]I would name some instead of just referring to the whole genre, as I've been trying to do generally in this discussion, but I genuinely couldn't tell you the name of any of them. They all have blended together in my head over the years. Generally they're something along the lines of *Hunie/Waifu Pop/Match/Quest* and then a number bigger than four but smaller than thirteen, and the cover art is an anatomically confusing anime-styled woman in a swimsuit or impractical armor.

References

Droettboom, John D. Hunter, Michael. n.d. "Matplotlib: Python Plotting Package."

Galyonkin, Sergey. 2021. "Steam Spy." *SteamSpy - All the Data about Steam Games*. https://www.steamspy.com/about.

Harris, Charles R., K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, et al. 2020. "Array Programming with NumPy." *Nature* 585 (7825): 357–62. https://doi.org/10.1038/s41586-020-2649-2.

Kluyver, Thomas, Benjamin Ragan-Kelley, Fernando Pérez, Brian Granger, Matthias Bussonnier, Jonathan Frederic, Kyle Kelley, et al. 2016. "Jupyter Notebooks a Publishing Format for Reproducible Computational Workflows." In *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, edited by F. Loizides and B. Schmidt, 87–90. IOS Press.

Kuchera, Ben. 2017. "Valve Exposes the Magic of Steam's Recommendation System, Shows It to Players." *Polygon*. https://www.polygon.com/2017/5/9/15591768/valve-steam-curators-reviews-friends.

Metacritic. n.d. "Disco Elysium." *Metacritic*. https://www.metacritic.com/game/pc/disco-elysium.

PCGamesN. n.d. "Steam's New Recommendation Algorithm Is a 'Catastrophe' for Some Indie Devs | PCGamesN." https://www.pcgamesn.com/steam/wishlists.

Pérez, Fernando, and Brian E. Granger. 2007. "IPython: A System for Interactive Scientific Computing." *Computing in Science and Engineering* 9 (3): 21–29. https://doi.org/10.1109/MCSE.2007.53.

Portugal, Ivens, Paulo Alencar, and Donald Cowan. 2018. "The Use of Machine Learning Algorithms in Recommender Systems: A Systematic Review." *Expert Systems with Applications* 97 (May): 205–27. https://doi.org/10.1016/j.eswa.2017.12.020.

Ransbotham, Shawn P. Curley, Jesse Bockstedt. n.d. "The Hidden Side Effects of Recommendation Systems." *MIT Sloan Management Review*. https://sloanreview.mit.edu/article/the-hidden-side-effects-of-recommendation-systems/.

"Scikit-Learn/Scikit-Learn." 2021. scikit-learn.

"Seatgeek/Fuzzywuzzy." 2021. SeatGeek.

Statista. n.d. "Number of Games Released on Steam 2020." *Statista*. https://www.statista.com/statistics/552623/number-games-released-steam/.

Steamworks. n.d. "Steamworks Partner Program." https://partner.steamgames.com/steamdirect.

Tamber. 2017. "Steam Video Games." https://kaggle.com/tamber/steam-video-games.

team, The pandas development. 2020. "Pandas-Dev/Pandas: Pandas." Zenodo. https://doi.org/10.5281/zenodo.3509134.

Van Rossum, Guido, and Fred L. Drake. 2009. *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace.

Wehrmeyer, Stefan, Gregor Aisch. n.d. "Dataset: Toolkit for Python-Based Database Access."