Steam Recommender Documentation

Abstract

Recommendation systems are used everywhere on the internet to help its users find things to enjoy. From shopping for items on Amazon to finding new songs to listen to on Spotify. And so, I decided to create my own recommendation system for finding games from Steam. To accomplish this, I use two different datasets full of game metadata and user information. I use the user-defined tags associated with the games on Steam from the metadata dataset. And with these tags I used cosine similarity to find the most similar games to the target game within a game dataset. And using these similar games I went through dataset of user information to find what games commonly show up alongside the target game with the top ten of these games being the recommendations that the system gives. The end result of this project is a steam recommendation system that I believe makes sensible suggestions based on the entered target game. Along the way I also do some simple dataset analysis to look for any interesting points.

Introduction

In most online marketplaces there is a section that the customer can look at to find products that the marketplace has decided may be of some interest to the customer. One example of this is Amazon, which uses a customer's previous purchase history to recommend new products to them. Other places that you can find these systems in place are streaming services like Netflix, Spotify, YouTube, etc. These places all have recommendation systems in place to help the customer find new shows or music to entertain themselves with. And For my Graduate capstone project I decided that I wanted to attempt and create a recommendation system of my own. And instead of making one for movies or music, I wanted to create one that would recommend video games based on an entered game. To that end, I will be using Steam, the largest digital distribution platform for pc games, as my marketplace of choice for creating game recommendations. My goal is to create a recommendation system that will take a target game, let's say a platformer like Mario, and using its tags and user information I want to make a list of sensible recommendations for the user.

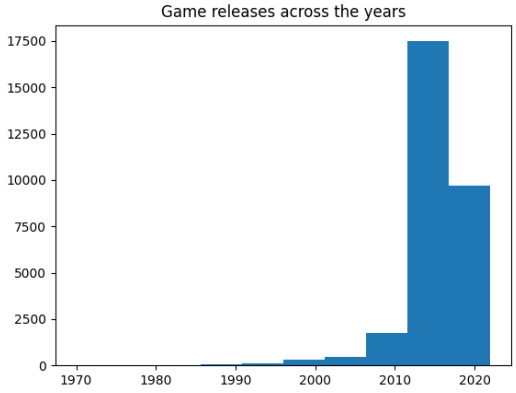
Datasets

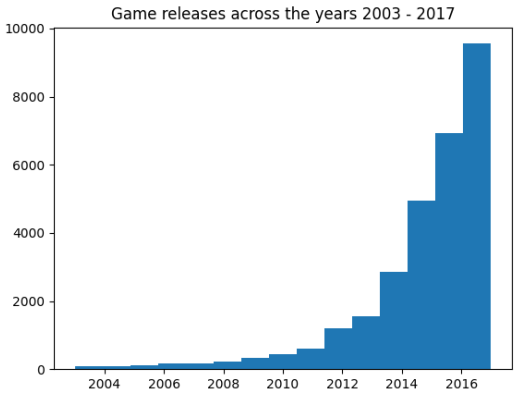
In my capstone I will be using two separate databases from Julian McAuley's recommender systems datasets. The first being a dataset of around 30,000 entries, each corresponding to the metadata for a specific game. This information includes the game's publisher, genres, name, URL, release date, price, developer, tags, and more. The second database holds information from around 80,000 users. And it consists of the user's id, how many games the user owns, and a list of every game that the user owns. The list of games was made up of the name of each game, and the number of hours that the user spent playing the game. A few

things to take note of for these datasets is that they were created 2-3 years ago so they are mildly outdated and that the user information dataset is comprised of data that was taken from Australian users.

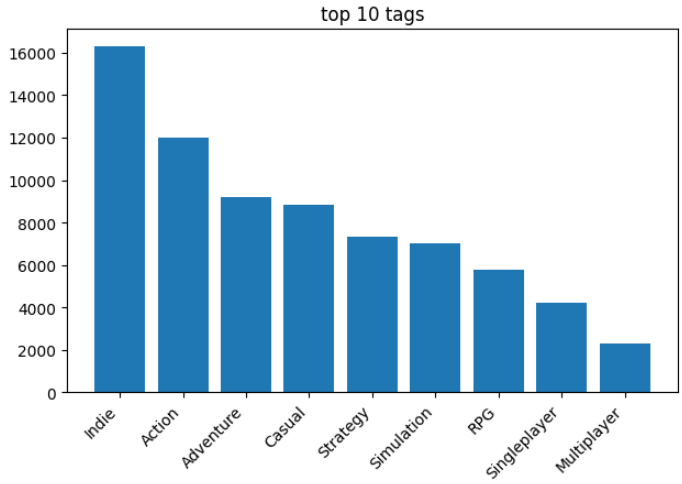
Dataset analysis

Before I get into how I went about creating my recommendation system I'm going to first do some analysis on the game metadata datasets to see what can be discovered. The first thing I'm going to is check is the release dates on from the metadata dataset. I want to see if there is a trend to the amount of games that are being released over time. Now when I did this the first time I saw that for the histogram went from 1970 to 2020, however steam has only been open since 2003. So, when I looked into this I noticed that aside from games, steam also sells movies and within the databases there are movies that were made in 1970 and afterward. To combat this, I looked through the tags and removed anything that had the movie or documentary tag. And for anything that was missed I also made a subset of the years from 2003 to 2017 which is when the majority of the games were released. And rom that we can see that the amount of games being released has been increasing as time passed.





Another thing that I'm curious about is which tags are the most common across all games. Since there are over 300 unique tags I will only be outputting the top 10 most common tags. The most common tag across all of the games is the indie tag, which are is a tag associated with games created by small development teams that don't have the backing of a large game publisher. Followed by action and adventure games which are two popular game genres. The least common of the top 10 tags is the multiplayer tag which is self-explanatory. I think this has been enough analysis for this and it's time to move on to the main topic.



Feature selection

I initially decided that I would include the game's developer, publisher, genres and tags into the recommender system. The developer and publisher variables since users do tend to enjoy games made or released by the same people. Genres since those are the basic categories used to classify games, which includes action, adventure, comedy, etc. And tags because these are user-defined descriptors for the game that give more information on what is in the game. Some common tags are male/female protagonist, story rich, and 2D/3D graphics. However, as I was looking through the data set I noticed that the categories that appeared in the genres variables also appeared in the tags variables since if a game is in the action genre, users would also apply the action tag to the game. Because of this, I decided to omit the genres variable from the system to avoid redundant information showing up. And as I was looking through the developer and

publisher variables I found that there were over 3000 entries in the database that had left these variables empty. So, I went ahead and omitted those variables as well because there were too many entries missing those variables to justify removing those entries entirely. This left me with only being able to use the tags variable as the basis for my recommendation system. The tags variables had plenty of information about what makes up the game in question. And the entries missing tags was a small enough portion of the dataset that I could just remove them without affecting the outcome of the overall system. However, using the tags variables brought its own set of issues that I will discuss later in this report.

On the other hand, I had a much easier time deciding what attribute I would be using from the user information database. As I stated earlier, this database was comprised of the user's ID, total amount of games owned, and a list of every owned game. Of these attributes the user's ID and the amount of games that they owned held no significance in what I was trying to accomplish with the recommender system. This left me with the list of every game that the user owns. To be more specific, only the names for each game that they owned was useful in the process of creating the game recommendations.

Program

For my capstone project I will be using python as my programming language of choice due to personal familiarity with the language and the ease with which I can use the required packages. I will primarily be using the pandas and sklearn python packages in my project. I am using the Pandas package to create the dataframes that I will be using throughout my program. And from the sklearn package, I am using the TfidfVectorizer to create a vector from a string that I will present to you later in this report. And I am also taking, arguably, the most important part of this project from the sklearn package, cosine similarity. Cosine similarity will take the

two vectors that I create through the vectorizer and compare them to see how similar they are which is an integral aspect for the recommendation process.

Cosine similarity works by finding the cosine of the angle between two vectors projected into multi-dimensional space. The outcome of cosine similarity lies within the range of [-1,1] where a result two proportional vectors will be 1 showing that they are very similar. Meanwhile, if the two vectors are perpendicular to each other they will have a cosine similarity of 0 meaning that the two vectors are very different to each other. A result of -1 means that the vectors show two opposing views and shows up if the vectors have an angle of 180 degrees. I chose cosine similarity as the basis for my project because it works well for sparse vectors which is important because there are some games that only have three tags while others would have eight which would make game have many zeros in the vector. Another similarity measurement that I could have used for this capstone is the Jaccard index, however I chose cosine similarity due to familiarity with it from previous classes that I have taken.

Pre-processing

The first thing that I had to do before I could start creating the recommendation system program was process the data. I started with the item metadata dataset and began by first removing any entries that were completely blank. Following that, I started looking for any entries that didn't have any titles. Once these had been found, I worked on filling in these missing titles. This was surprisingly easy to do since each entry in the dataset had an app name variable and a title variable, and I noticed that these variables were the exact same for every entry that contained both an app name and a title. So, I went through the dataset and for every game that didn't have a title, but did have an app name, I would put the app name into the title variable. However, for the entries that didn't have either an app name or title, I used the URL for the game

that was included in the dataset and manually found the name and entered it in. If this was not possible, due to the URL not working, I had to drop the entry because I couldn't get the name of the game.

The last thing I had to go through was the tags to find any that didn't have any tags. Which turned out to be about 160 entries, I ended up dropping these entries due to this being a very small portion of the total data in the dataset. And I considered imputing the missing the data however I ultimately decided against it. Since imputing the tags would just cause the affected entries to be represented incorrectly. Which would just cause the game to appear as an average of the most used tags which would just lead to said imputed games appearing in recommendations for games that they shouldn't be in and lowering the chances for games that should actually be recommended for the target search from appearing.

Compared to the item metadata dataset, the user information dataset didn't have that many things that I needed to clean regarding the user information dataset. The only things I had to check were if there were any entries that were blank. If so, I went ahead and dropped them since I couldn't get the information to fill that in. And the other criteria I looked into was whether the user owned less than 2 games. These entries wouldn't help with recommender system, so I went ahead and dropped them. There wasn't anything else that needed to be checked in this data set.

Recommendation process

For the system to create suitable recommendations the user must first enter in a target game, for the sake of this report let's say the user enters in Portal as the target game. And using Portal, the system then goes through the metadata dataset and retrieves its corresponding entry

and the system extracts the game's list of tags from it. The system then transforms said list of tags from Portal into a comma separated string which will be used in the next step. Using the string of tags, we proceed to use the TfidfVectorizer from the sklearn python package and convert it into a vector. Before we can move onto the next step, the program goes through the metadata dataset and for every game extracts their tags and does the same transformation for each one that we did for the target game. Then, once the tags for every entry in the metadata dataset has been turned into a vector I use cosine similarity to compare the tags vector of the target game, in this case Portal, to the tags vector that was created for every other game in the metadata dataset. The recommendation system than ranks each entry by how similar their tags vector are to the target game's tags vector.

Once the ranking has been accomplished the system takes the top fifty most similar games and moves on to the second step of the recommendation process. This step requires the use of information from the user dataset. Specifically, it requires using the list of all games that the users own. Using the top fifty most similar games, the recommender iterates through the user database and looks for users that own the target game. When the system finds a user that meets that criteria, it then looks to see if any of the top similar games appear alongside the target game in said user's game library. While doing this, the program keeps a running count of the number of times each similar game appears alongside the target game. In the final step of the process, the recommender sorts the fifty similar games by how many times they appear alongside the target game and outputs the top 10 games as its recommendation for the user.

Performance

In the case of Portal, which I stated as the target game for this project, we see that the program recommends Portal 2 and several half life games Which were all made by Valve, the developer that also created Portal. Following that there is a Portal Stories: Mel. A community made mod for portal 2 that also has overwhelmingly positive reviews. I'm not sure why Thomas was Alone, Bully, little Inferno, or Tiny and Big were recommended for this target game, I think it probably has to do with how outdated parts of the dataset is. However, they are still great games to play nonetheless.

```
Target game: Portal
Portal 2
Half-Life 2
Half-Life 2: Episode One
Half-Life: Opposing Force
Half-Life: Blue Shift
Thomas Was Alone
Little Inferno
Bully: Scholarship Edition
Portal Stories: Mel
Tiny and Big: Grandpa's Leftovers
```

Another game that I ran through the program was Deponia, a point and click adventure game. We can see that the top 2 games that the program recommends other games from the Deponia franchise, Chaos on Deponia and Goodby Deponia. And outside of those games, A New Beginning – Final cut was made by the same developer as the target game. Following those two games all of the other recommendations made by the program are all point and click adventure games that have positive reviews on steam. And aside from the presented examples I also tested several other games and gained similarly decent results.

```
Target game: Deponia
Chaos on Deponia
Goodbye Deponia
Little Inferno
A New Beginning - Final Cut
Machinarium
Broken Age
The Book of Unwritten Tales
The Journey Down: Chapter One
Samorost 2
AR-K
```

And after looking at the results given by the program in terms of these two examples I feel mostly satisfied with how the it performs. Because the main criteria that I was looking at to judge whether the recommendation system works was whether it displayed games that followed the same concept, i.e. recommending action games for action target games. And whether the games getting recommended are made by the same developer since, as I stated earlier in the

feature selection section, people usually enjoy games made by the same developers. It was a bonus if the recommended games also had positive reviews, but it wasn't required. And looking at the results for the two examples above I would say that recommender fulfilled these criteria.

Problems

I would say that the biggest problem that I have faced while creating this system was the nature of steam tags themselves. Tags are user-defined identifiers for the game and, as you can imagine, due to the fact that they are user-defined, they are prone to being updated over time as more users assign tags to the game. For example, if we compare the tags for Portal 2 from the database and from the current steam marketplace we'll see that there several major changes between the two. Another characteristic of tags that had to be dealt with was that they aren't always as in-depth as I would have like them to be when creating my recommender. This caused me to question the performance of the program and whether I needed to make changes to recommender. Because while running Portal as the target game will have the recommender bring up Portal 2. Having Portal 2 as the target game does not bring up Portal as a recommendation. And this was due to just how generic the tags were for the Portal 2 entry in the database.

The other issues that I had were relatively minor in comparison. With one of them being just the sheer size of the data sets themselves. I couldn't just read them in since they were too big and instead had to use a function that I found on the website where I found these data sets that would pull a single line from the dataset at a time and append that to an empty list. Because of how long this would take, every time I ran the program I would have to wait 2-3 minutes before the results would appear. Because of this whenever I made a mistake in entering in the name of the target game I would have to wait 2-3 minutes before I could try again which wasn't that bad

but it builds up over several attempts. The other minor issue that I had was just that the databases that I am using are outdated. As far as I can tell both datasets were created sometime around 2017-2018. This just made it so that I had to double check to make sure that the target game that I was using did indeed exist within the metadata dataset. Other than that there wasn't really anything else that caused me to have problems. Creating this project was fairly straightforward and not incredibly complicated.

Improvements

If I were to continue working on this recommendation system some there are a few things that I would change in hopes of improving the recommendations that the program makes and help make it easier to use. The first improvement being updating the tags for each game so because they have matured quite a bit in the time since the creation of these datasets. I would like to create a simple user interface that would that would accept input and then output the recommended games along with their cover art to make them easier to identify instead of just outputting a simple list of names. Possibly making it so that clicking on the images would allow the user to redirected to Steam's listing for the recommended game.

conclusion

In conclusion, Recommendations systems are used in many different places throughout the internet, from Amazon suggesting new products to Netflix and Spotify for find new shows and music to stream. And for my capstone project I created a game recommendation system. With Steam as my marketplace of choice since it is the largest digital distribution platform for pc games. My goal with this project was to create a recommender that takes the tags from a user-specified target game. And using those tags, create a list of 10 or so games that the user could

possible enjoy. I accomplished this through the use of cosine similarity to find games with similar tags as those of the target game. And following that I loooked through user libraries to see which ones appeared alongside the target game the most. And while there were a few issues that I had to deal with throughout the creation of this project, the resultant recommendation system that I have created gives satisfactory recommendations that are sensible given the target game.

Refrences

McAuley, Julian. “Recommender Systems and Personalization Datasets.” *Recommender Systems Datasets*, https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data.

https://towardsdatascience.com/building-a-game-recommendation-engine-870a1ccd11c4

Self-attentive sequential recommendation
Wang-Cheng Kang, Julian McAuley
ICDM, 2018
pdf

Item recommendation on monotonic behavior chains
Mengting Wan, Julian McAuley
RecSys, 2018
pdf

Generating and personalizing bundle recommendations on Steam
Apurva Pathak, Kshitiz Gupta, Julian McAuley
SIGIR, 2017
pdf