CSC111 Project Review: Intelligent recommendation by item-based collaborative filtering algorithm: Want More Games and Friends?

Yanke Mao, Weiheng Wang, Yiteng Sun, Jiaxu Li

Wednesday, April 14, 2021

Introduction

As a citizen living in an age of data and technology. A majority of us have benefited from the convenience and efficiency of all kinds of apps on our phones. Sometimes we would wonder that after sometime of using, your phone seems to know you better. From where you would like to eat for lunch, to what to buy for a friend's birthday. Aside from your family and closest friends, these apps start to step into your life deeper an deeper(Nicole Spector, 2019). The Algorithms of all these apps all have one thing in common – Keep pushing new contents based on what your have browsed in your history. Also known as the 'You might like' Algorithms. By implementing these Algorithms, the users would have a customized experience on the apps they use(Mary Aleksandrova, 2018). Inspired by the everyday apps we use, we noticed that in the game fields, especially for video game lovers, which is a huge crowd nowadays, always have trouble finding the game that worth purchasing and devoting efforts.(Tao, Z., Wei, Y., Wang, X., 2020) Stepping on a bomb game is what a lot of gamers are experiencing every day. A recommendation that is truly reliable and trustworthy is desperately needed for gamers. However, the famous game platform 'Steam' has already provided some approaches to help solve this problem – Wish list, You might like, and so on. What we are trying to achieve here is to imitate what steam has done so far: **We will provide a collection of games or users for recommendation using the knowledge of graphs and item-based collaborative filtering method, based on the data of games of similar category, and finally create an interface for users to do recommendation automatically.**

Dataset description

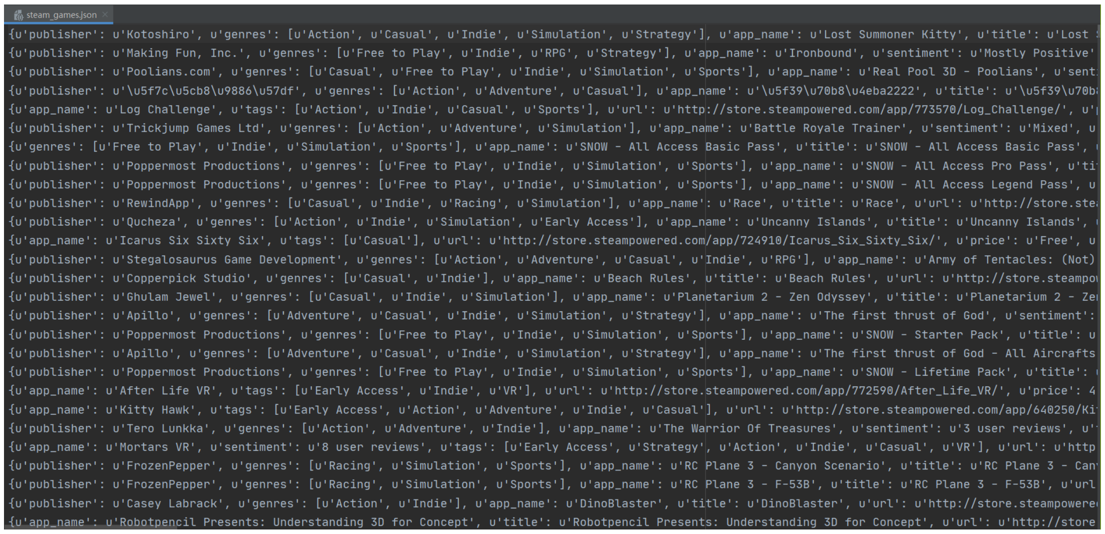
In the project we worked on, we have used two datasets for our computation: australian_user_reviews and steam_games. It is all in .json format.

australian_user_reviews.json: This dataset is what we have obatained from steam.com.It includes the recommendation and the information of the user who is rating on a certain game. Such as user_id, user_url, review status(which includes some brief comments and the posting status), recommendation(true or false represents the final opinion of the user towards the game). We will mainly focus on the recommendation, which is the 6th colunmn of the review section, as the basis of judging whether to recommend or not.

steam_games.json: This dataset is also collected from steam.com. it includes but not limited to the status of a game: name, url of the game, publisher, genres and tags, price and discounted price, id of the game. This dataset i s used for filtering the games for recommendation.

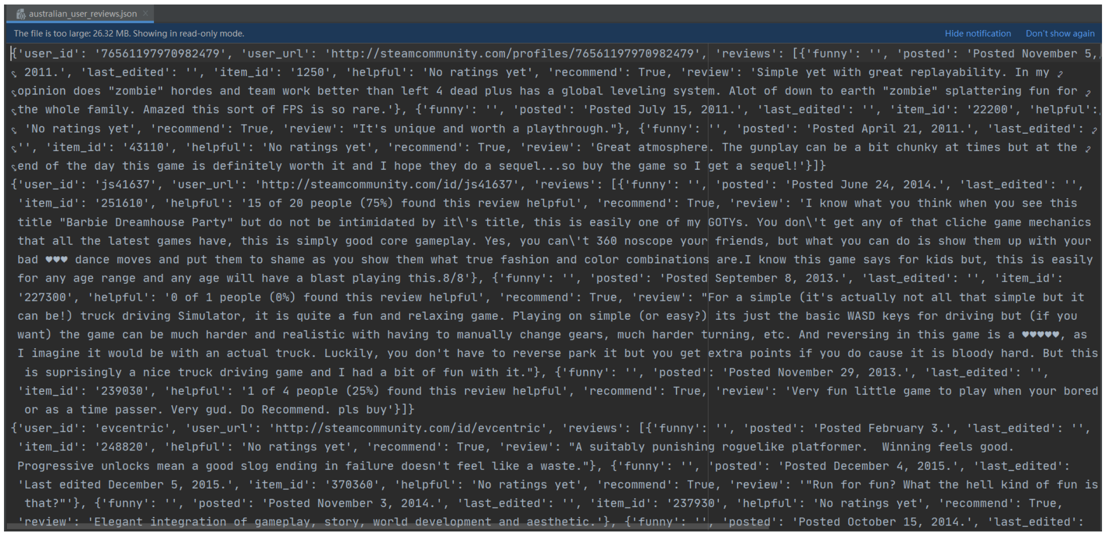
Computational Overview

In this project, we are going to use two main datasets: australian_user_reviews.json and steam_games.json to recommend proper games to user. The first data set we are going to use is the steam games data, which is shown in the attachment *australian_user_reviews.json*(see Figure 2 below). And we are going to store this data in our game recommendation graph. In this graph, each vertex can represent different object. We will use the same method in Assignment 3 to classify two distinct kinds of objects: 'user' represents each player and 'game' represents each game, and the "item" attribute can be 'user_id' or 'game_name'. The edge will exist between two vertex if and only if 'user' gave the comment to this 'game'. In this way, after import our user reviews data, we can obtain a graph about the relationship between the user and the game. And the second data we are going to use is the steam games data, like the data shown in *steam_games.json*(see Figure 1 below). For this dataset, we mainly want to use the 'game_id' we took from the *australian_user_items.json* to search for the detailed information about this game in the dataset. Then we will use the functions to read this two datasets and filter the information we don't need. Then we can get two dictionaries: one is the user_reviews data, the format is {'user_id': ([{'game_id': 'recommend'},...], 'user_url')}, and the other one is the steam_games data, which the format of the data is {'game_id': ('title', 'url')}. In the next process, we will run the function load_weighted_graph to store the data in our game recommendation graph.



```
{u'publisher': u'Kotoshiro', u'genres': [u'Action', u'Casual', u'Indie', u'Simulation', u'Strategy'], u'app_name': u'Lost Summoner Kitty', u'title': u'Lost
{u'publisher': u'Making Fun, Inc.', u'genres': [u'Free to Play', u'Indie', u'RPG', u'Strategy'], u'app_name': u'Ironbound', u'sentiment': u'Mostly Positive'
{u'publisher': u'Poolians.com', u'genres': [u'Casual', u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'Real Pool 3D - Poolians', u'sent
{u'publisher': u'\u5f7c\u5cb8\u9886\u57df', u'genres': [u'Action', u'Adventure', u'Casual'], u'app_name': u'\u5f39\u70b8\u4eba2222', u'title': u'\u5f39\u70b
{u'app_name': u'Log Challenge', u'tags': [u'Action', u'Indie', u'Casual', u'Sports'], u'url': u'http://store.steampowered.com/app/773570/Log_Challenge/', u'
{u'publisher': u'Trickjump Games Ltd', u'genres': [u'Action', u'Adventure', u'Simulation'], u'app_name': u'Battle Royale Trainer', u'sentiment': u'Mixed', u
{u'genres': [u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'SNOW - All Access Basic Pass', u'title': u'SNOW - All Access Basic Pass',
{u'publisher': u'Poppermost Productions', u'genres': [u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'SNOW - All Access Pro Pass', u'ti
{u'publisher': u'Poppermost Productions', u'genres': [u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'SNOW - All Access Legend Pass', u
{u'publisher': u'RewindApp', u'genres': [u'Casual', u'Indie', u'Racing', u'Simulation'], u'app_name': u'Race', u'title': u'Race', u'url': u'http://store.ste
{u'publisher': u'Qucheza', u'genres': [u'Action', u'Indie', u'Simulation', u'Early Access'], u'app_name': u'Uncanny Islands', u'title': u'Uncanny Islands',
{u'app_name': u'Icarus Six Sixty Six', u'tags': [u'Casual'], u'url': u'http://store.steampowered.com/app/724910/Icarus_Six_Sixty_Six/', u'price': u'Free', u
{u'publisher': u'Stegalosaurus Game Development', u'genres': [u'Action', u'Adventure', u'Casual', u'Indie', u'RPG'], u'app_name': u'Army of Tentacles: (Not)
{u'publisher': u'Copperpick Studio', u'genres': [u'Casual', u'Indie'], u'app_name': u'Beach Rules', u'title': u'Beach Rules', u'url': u'http://store.steampo
{u'publisher': u'Ghulam Jewel', u'genres': [u'Casual', u'Indie', u'Simulation'], u'app_name': u'Planetarium 2 - Zen Odyssey', u'title': u'Planetarium 2 - Ze
{u'publisher': u'Apillo', u'genres': [u'Adventure', u'Casual', u'Indie', u'Simulation', u'Strategy'], u'app_name': u'The first thrust of God', u'sentiment':
{u'publisher': u'Poppermost Productions', u'genres': [u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'SNOW - Starter Pack', u'title': u
{u'publisher': u'Apillo', u'genres': [u'Adventure', u'Casual', u'Indie', u'Simulation', u'Strategy'], u'app_name': u'The first thrust of God - All Aircrafts
{u'publisher': u'Poppermost Productions', u'genres': [u'Free to Play', u'Indie', u'Simulation', u'Sports'], u'app_name': u'SNOW - Lifetime Pack', u'title':
{u'app_name': u'After Life VR', u'tags': [u'Early Access', u'Indie', u'VR'], u'url': u'http://store.steampowered.com/app/772590/After_Life_VR/', u'price': 4
{u'app_name': u'Kitty Hawk', u'tags': [u'Early Access', u'Action', u'Adventure', u'Indie', u'Casual'], u'url': u'http://store.steampowered.com/app/640250/Ki
{u'publisher': u'Tero Lunkka', u'genres': [u'Action', u'Adventure', u'Indie'], u'app_name': u'The Warrior Of Treasures', u'sentiment': u'3 user reviews', u'
{u'app_name': u'Mortars VR', u'sentiment': u'8 user reviews', u'tags': [u'Early Access', u'Strategy', u'Action', u'Indie', u'Casual', u'VR'], u'url': u'http
{u'publisher': u'FrozenPepper', u'genres': [u'Racing', u'Simulation', u'Sports'], u'app_name': u'RC Plane 3 - Canyon Scenario', u'title': u'RC Plane 3 - Can
{u'publisher': u'FrozenPepper', u'genres': [u'Racing', u'Simulation', u'Sports'], u'app_name': u'RC Plane 3 - F-53B', u'title': u'RC Plane 3 - F-53B', u'url
{u'publisher': u'Casey Labrack', u'genres': [u'Action', u'Indie'], u'app_name': u'DinoBlaster', u'title': u'DinoBlaster', u'url': u'http://store.steampowere
{u'app_name': u'Robotpencil Presents: Understanding 3D for Concept', u'title': u'Robotpencil Presents: Understanding 3D for Concept', u'url': u'http://store
```

Figure 1: sample steam_games.json data



The file is too large: 26.32 MB. Showing in read-only mode.

```
{'user_id': '76561197970982479', 'user_url': 'http://steamcommunity.com/profiles/76561197970982479', 'reviews': [{'funny': '', 'posted': 'Posted November 5,
 2011.', 'last_edited': '', 'item_id': '1250', 'helpful': 'No ratings yet', 'recommend': True, 'review': 'Simple yet with great replayability. In my
opinion does "zombie" hordes and team work better than left 4 dead plus has a global leveling system. Alot of down to earth "zombie" splattering fun for
the whole family. Amazed this sort of FPS is so rare.'}, {'funny': '', 'posted': 'Posted July 15, 2011.', 'last_edited': '', 'item_id': '22200', 'helpful':
 'No ratings yet', 'recommend': True, 'review': "It's unique and worth a playthrough."}, {'funny': '', 'posted': 'Posted April 21, 2011.', 'last_edited':
'', 'item_id': '43110', 'helpful': 'No ratings yet', 'recommend': True, 'review': 'Great atmosphere. The gunplay can be a bit chunky at times but at the
end of the day this game is definitely worth it and I hope they do a sequel...so buy the game so I get a sequel!'}]}
{'user_id': 'js41637', 'user_url': 'http://steamcommunity.com/id/js41637', 'reviews': [{'funny': '', 'posted': 'Posted June 24, 2014.', 'last_edited': '',
 'item_id': '251610', 'helpful': '15 of 20 people (75%) found this review helpful', 'recommend': True, 'review': 'I know what you think when you see this
 title "Barbie Dreamhouse Party" but do not be intimidated by it\'s title, this is easily one of my GOTYs. You don\'t get any of that cliche game mechanics
 that all the latest games have, this is simply good core gameplay. Yes, you can\'t 360 noscope your friends, but what you can do is show them up with your
 bad ♥♥♥ dance moves and put them to shame as you show them what true fashion and color combinations are.I know this game says for kids but, this is easily
 for any age range and any age will have a blast playing this.8/8'}, {'funny': '', 'posted': 'Posted September 8, 2013.', 'last_edited': '', 'item_id':
 '227300', 'helpful': '0 of 1 people (0%) found this review helpful', 'recommend': True, 'review': "For a simple (it's actually not all that simple but it
 can be!) truck driving Simulator, it is quite a fun and relaxing game. Playing on simple (or easy?) its just the basic WASD keys for driving but (if you
 want) the game can be much harder and realistic with having to manually change gears, much harder turning, etc. And reversing in this game is a ♥♥♥♥♥, as
 I imagine it would be with an actual truck. Luckily, you don't have to reverse park it but you get extra points if you do cause it is bloody hard. But this
  is suprisingly a nice truck driving game and I had a bit of fun with it."}, {'funny': '', 'posted': 'Posted November 29, 2013.', 'last_edited': '',
 'item_id': '239030', 'helpful': '1 of 4 people (25%) found this review helpful', 'recommend': True, 'review': 'Very fun little game to play when your bored
  or as a time passer. Very gud. Do Recommend. pls buy'}]}
{'user_id': 'evcentric', 'user_url': 'http://steamcommunity.com/id/evcentric', 'reviews': [{'funny': '', 'posted': 'Posted February 3.', 'last_edited': '',
 'item_id': '248820', 'helpful': 'No ratings yet', 'recommend': True, 'review': "A suitably punishing roguelike platformer.  Winning feels good.
 Progressive unlocks mean a good slog ending in failure doesn't feel like a waste."}, {'funny': '', 'posted': 'Posted December 4, 2015.', 'last_edited':
 'Last edited December 5, 2015.', 'item_id': '370360', 'helpful': 'No ratings yet', 'recommend': True, 'review': '"Run for fun? What the hell kind of fun is
  that?"'}, {'funny': '', 'posted': 'Posted November 3, 2014.', 'last_edited': '', 'item_id': '237930', 'helpful': 'No ratings yet', 'recommend': True,
 'review': 'Elegant integration of gameplay, story, world development and aesthetic.'}, {'funny': '', 'posted': 'Posted October 15, 2014.', 'last_edited':
```

Figure 2: sample australian_users_reviews data

In our computation part, we are going to use a simplified item-based collaborative filtering method to do a recommendation for a user. Since the item-based algorithms provide a better performance and quality than user-based algorithms, and at the same time, it is easier to scale to large data set (B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl, 2001), we select the item-based technique in our project.

There are two ways to do the recommendation. One is doing recommendation for our certain user. In this part, first we will use the concept of matrix and store the rating data for each game in matrix. Each row represents a user, and each column represents a certain games. We define that if a user recommends this game ('recommend': 'True'), then the his rating to this game is 1. If a user does not recommend it ('recommend': 'False'), then we set the rating with 0. For the user does not make any comments to this game, we set the rating with 'N/A'. After that, we are going to calculate the Cosine-based Similarity between these users and our certain user. First, we will calculate the numerator. If the two users both gave their comments to one game, then we multiply the two ratings and add all of them. Next, we will calculate the denominator, using the formula $\sqrt{\sum_{i=1}^{n} A_i^2} * \sqrt{\sum_{i=1}^{n} B_i^2}$ where A and B represent the rating of user A and the rating of user B respectively. With the definition of Cosine-based Similarity, sim(i, j) = cos(i, j) (B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl, 2001). And the bigger the Cosine-based Similarity is, the more similar the related game is to the matching game. Finally, we can obtain the Cosine-based Similarity for every user. And we are going to recommend the players of top 5 cosine similarity to the user to play together.

The second way is to recommend related games based on a certain game. In this way, we are going to select the related games for this game and calculate their Cosine-based Similarity and recommend the games with top5 similarity.The method to calculating the Cosine-based Similarity is same to the above.In this part, each row represents a certain game and each column represents a certain user.

To show report the result of our project, we used a new library tkinter to construct an interface for users to recommend games based on their favorite games and game friends based on their reviews of games. In the interface, we use the tkinter.Radiobutton() function to construct two options for users to freely choose which recommendation they want us to perform. They can choose either to enter their user id to obtain recommended new friends or to enter the name of their favorite games to obtain recommended games. And under these two options, we use tkinter.Entry() function to provide users a textbox to type the game name or their user id. Besides, we created a button on the bottom of the interface. Once users press the button, our program will automatically generate a recommendation list for users. And the result of the recommendation will be shown in a new pop-up window constructed by the function tkinter.messagebox. If the app name/user id user entered is in our data, the recommended friends/games will appear together with their URL in the pop-up window; if the app name/user id user entered is not in our data sets, there will be a warning pop-up window tell users the input id is not in the library. And if the game name is in the data sets but unluckily there aren't any similar games in our library, the pop-up window will tell users this disappointing information.

Instruction

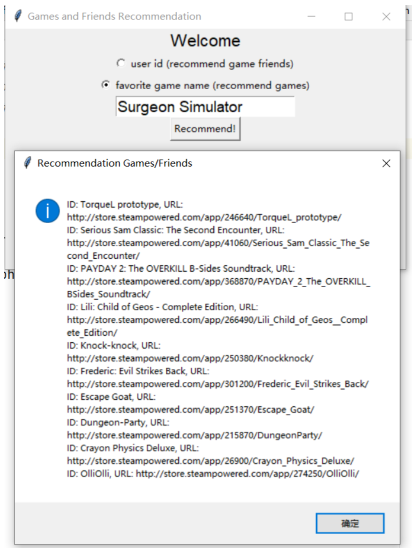
To run our program, first, you need to install the new libraries listed in the attachment requirements.txt. And you should download the data sets called 'australian_user_reviews.json' and 'steam_games.json' which we have uploaded them through UTsend. After that, you can run the file main.py in our program.

Our program will produce an interface with the title 'Games and Friends Recommendation' and a label 'Welcome'. Below the label, there are two options you can choose to do a recommendation. The first one is 'user id'. When you choose this option, you should enter your user id in the entry box and pressing the 'Recommend!' button below to start recommendation for game friends. The second is 'favorite game name'. when choosing this you should enter your favorite game name and pressing the 'Recommend!' button below to start recommendation for similar games. Notice you can only choose one way to do a recommendation at one time. The result of the recommendation will be shown in a new pop-up window. Since our data is limited and the identification of games' names is strict (especially for Letter case), some inputs may return a warning pop-up window telling that the input id is not in the library. Thus, we provide some valid example input of games' names and user id. We strongly suggest you enter these values to test our program.

Valid games' name:
{'Indie Game: The Movie', 'Kinetic Void', 'Surgeon Simulator', 'Star Trek Online', 'The Guild II Renaissance', 'Train Simulator', 'Enemy Front', 'Borderlands 2: Mr. Torgue's Campaign of Carnage', 'Roguelands', 'E.Y.E: Divine Cybermancy', 'Fallout 4 - Wasteland Workshop', 'Tree of Savior (English Ver.)', 'Orcs Must Die! 2 - Family Ties Booster Pack', 'Atom Zombie Smasher', 'Faerie Solitaire', 'Iron Grip: Warlord', 'RollerCoaster Tycoon World™', 'Vector', 'Dead State: Reanimated', 'Picross Touch'}

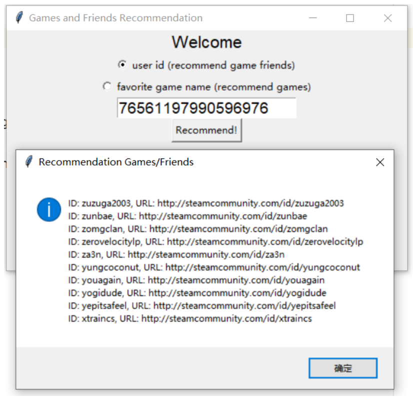
Valid user id:
{'76561198076144807', '76561197990596976', '76561198009729086', '76561198056913746', 'Tezzdog', '76561198046170033', 'n3rdra93', 'lilbkbx', '76561198090455022', '-I_AM_EPIC-', 'GirlzLikeBears', '76561198062450095', 'hiimagay', '765611981033623

’gryphon556’, ’76561198084577659’, ’xxxzetaxxx’, ’AnimeHex’, ’playergdgsd’, ’jamesreesonswingspanis6ft’, ’preps427’, ’76561198067960387’, ’76561198068160202’, ’76561198179668470’, ’epic_doom’, ’76561198078812809’, ’seukuka’, ’765611980459001
’Cameross’, ’jefferyjefferson’, ’76561198074416924’, ’LordOfOreos’, ’76561198088045830’, ’76561198069902636’, ’robin-isamazing’}


Games and Friends Recommendation
Welcome
user id (recommend game friends)
[✓ favorite game name (recommend games)]
Surgeon Simulator
Recommend!
Recommendation Games/Friends
确定

ID: TorqueL prototype, URL: http://store.steampowered.com/app/246640/TorqueL_prototype/
ID: Serious Sam Classic: The Second Encounter, URL: http://store.steampowered.com/app/41060/Serious_Sam_Classic_The_Second_Encounter/
ID: PAYDAY 2: The OVERKILL B-Sides Soundtrack, URL: http://store.steampowered.com/app/368870/PAYDAY_2_The_OVERKILL_BSides_Soundtrack/
ID: Lili: Child of Geos - Complete Edition, URL: http://store.steampowered.com/app/266490/Lili_Child_of_Geos__Complete_Edition/
ID: Knock-knock, URL: http://store.steampowered.com/app/250380/Knockknock/
ID: Frederic: Evil Strikes Back, URL: http://store.steampowered.com/app/301200/Frederic_Evil_Strikes_Back/
ID: Escape Goat, URL: http://store.steampowered.com/app/251370/Escape_Goat/
ID: Dungeon-Party, URL: http://store.steampowered.com/app/215870/DungeonParty/
ID: Crayon Physics Deluxe, URL: http://store.steampowered.com/app/26900/Crayon_Physics_Deluxe/
ID: OlliOlli, URL: http://store.steampowered.com/app/274250/OlliOlli/

Figure 3: final result of recommend games


Games and Friends Recommendation
Welcome
[✓ user id (recommend game friends)]
favorite game name (recommend games)
76561197990596976
Recommend!
Recommendation Games/Friends
确定

ID: zuzuga2003, URL: http://steamcommunity.com/id/zuzuga2003
ID: zunbae, URL: http://steamcommunity.com/id/zunbae
ID: zomgclan, URL: http://steamcommunity.com/id/zomgclan
ID: zerovelocitylp, URL: http://steamcommunity.com/id/zerovelocitylp
ID: za3n, URL: http://steamcommunity.com/id/za3n
ID: yungcoconut, URL: http://steamcommunity.com/id/yungcoconut
ID: youagain, URL: http://steamcommunity.com/id/youagain
ID: yogidude, URL: http://steamcommunity.com/id/yogidude
ID: yepitsafeel, URL: http://steamcommunity.com/id/yepitsafeel
ID: xtraincs, URL: http://steamcommunity.com/id/xtraincs

Figure 4: final result of recommend game friends

Discussion

After the computational step in our program. We have finally constructed an interface to recommend game friends which might have the same taste of game like you. And also recommend games that are similar to the previous ones you have played. Back to the main topic. Our solution of the computational research did answer the question we proposed in the proposal: We will provide a collection of games for recommendation and create an interface for the user to use in an interactive way. But out target was not completely successful. When extracting the data from steam, the extracting process went smooth but the data itself had some problems: Either the data of the game or the data of the users is too cumbersome as it includes too much information that we don’t need, such as the status of the user’s log in and last visited comments. We spend a lot of efforts on ‘cleaning up’ the data and turn it into the version that we could utilize. After the process of data processing, we met another problem: The status in the original data which implies the user recommend this game or not is too simple, true or false, we were expecting a more complicated judging system, for instance, a rating system from 1 to 10, this would make our recommendation more reliable since it has more levels and the similarity between two users or games would be more accurate. One more thing to point out, since the game names are of all kind and even sometimes involves emoji. It was very likely to cause an error in our name-picky program. We decided to use game id as the key for searching instead of the name of the game. That way we could reduce a lot of errors. Anyway, despite these obstacles and flaws we have met. We still come up with a recommendation of games and users you might be interested in. For further improvement, we think that we could improve the rating system in order to get a more reliable result. Also, for the interface part, we could convert the ID of the game back to the name of the game since the interface is for client to read. They certainly don't want to face a bunch of random IDs.

References

A. B.M. Sarwar, G. Karypis, J.A. Konstan, and J. Reidl. Item-based collaborative filtering recommendation algorithms. In Proceedings of the 10th International World Wide Web Conference, pages 285-295, 2001.

B. Mary Aleksandrova, HOW TO CREATE A GREAT USER EXPERIENCE: FRESH APP PERSONALIZATION IDEAS FOR ANY INDUSTRY, Eastern Peak, 2018, https://easternpeak.com/blog/how-to-create-great-a-user-experie

C. Nicole Spector, Why Mobile Games Are So Addicting - And How You Reclaim Your Time, Better by Today, 2019, https://www.nbcnews.com/better/lifestyle/why-mobile-games-are-so-addicting-how-reclaim-your-time-ncna10312

D. Tao, Z., Wei, Y., Wang, X., He, X., Huang, X., & Chua, T. (2020). MGAT: Multimodal Graph attention network for recommendation. Information Processing & Management, 57(5), 102277. doi:10.1016/j.ipm.2020.102277

E. Wang, C. (2020, June 07). Why TikTok made its user So Obsessive? The AI algorithm that got you hooked. Retrieved March 16, 2021, from https://cutt.ly/6zNV5gW

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

G. https://docs.python.org/3/library/tkinter.html

H. https://www.w3schools.com/python/numpy_intro.asp