CSC111 Project Proposal: Interactive Personalized Game Recommender

Yucheng Duan, Xiaoshuai Liu, Zhipeng Chen

Tuesday, March 16, 2021

Problem Description and Research Question

Video games have become an integral part of modern entertainment, with millions of players worldwide engaging with various gaming platforms. With the rise of digital game distribution platforms such as Steam, Epic Games Store, and PlayStation Network, players now have access to an overwhelming number of games. However, finding the next game to play remains a challenging task due to the sheer volume of available titles and the diverse preferences of players.

Traditional game recommendation systems rely on collaborative filtering, content-based filtering, or hybrid approaches. While these methods have been effective to some extent, they often fail to capture the complex relationships between games, users, and their playing habits. Moreover, many recommendation systems lack interactivity, the players may spend a lot of time rolling the scroll of their mouse when looking for new games.

The goal of this project is to **develop an interactive game recommendation system that allows users to input their previously played games and receive personalized game suggestions based on graph-based relationships and hierarchical game categories.**

This project matters because players often struggle to discover new games that align with their preferences. By leveraging Graph and Tree structures, we can improve the accuracy and personalization of game recommendations. Additionally, to enhance user engagement, we plan to incorporate 3D visualizations of the game graph, offering a novel way to explore game connections in an immersive and dynamic manner.

By combining structured data representation with interactivity, our system aims to revolutionize game discovery, making recommendations more intuitive, engaging, and user-driven.

Computational Plan

Data Representation

Our project will use real-world data from **Steam user interactions**, including user reviews, playtime, and game metadata. Specifically, the datasets include:

- `users_reviews.json`: Contains `user_id`, `item_id` (game), `recommend` (boolean), review text, and other relevant details.
- `users_items.json`: Contains `user_id`, `item_id`, `playtime_forever` (total playtime in minutes), and additional gameplay statistics.
- `bundle_data.json`: Provides game `item_id`, `genre`, pricing/bundle information, and related metadata.

A sample data entry from `users_reviews.json` looks like:


```
{
    "user_id": "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 teamwork better than Left 4 Dead plus has a global leveling
         system. A lot 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!"}
    ]
}
```


A sample data entry from `users_items.json` looks like:


```
{
    "user_id": "evcentric",
    "items_count": 137,
    "steam_id": "76561198007712555",
    "user_url": "http://steamcommunity.com/id/evcentric",
    "items": [
    {"item_id": "1200",
     "item_name": "Red Orchestra: Ostfront 41-45",
     "playtime_forever": 923,
     "playtime_2weeks": 0},
    {"item_id": "1250",
     "item_name": "Killing Floor",
     "playtime_forever": 1646,
     "playtime_2weeks": 0},
    {"item_id": "550",
     "item_name": "Left 4 Dead 2",
     "playtime_forever": 82,
     "playtime_2weeks": 0}
    ]
}
```


A sample data entry from `bundle_data.json` looks like:


```
{
    "bundle_final_price": "$66.46",
    "bundle_url": "http://store.steampowered.com/bundle/450/",
    "bundle_price": "$73.86",
    "bundle_name": "Dharker Studio 2015 Complete",
    "bundle_id": "450",
    "items": [
    {"genre": "Adventure, Indie, RPG",
```

```
    "item_id": "326950",
    "discounted_price": "$8.99",
    "item_url": "http://store.steampowered.com/app/326950",
    "item_name": "Sword of Asumi"},
   {"genre": "Adventure, Casual, Indie",
    "item_id": "348540",
    "discounted_price": "$12.99",
    "item_url": "http://store.steampowered.com/app/348540",
    "item_name": "Divine Slice of Life"}
   ]
}
```


Graph and Tree Representation

1. Graph (User-Game Interaction Graph)

- **Nodes**: Users (`user_id`) and games (`item_id`)
- **Edges**:
  - Between users and games (`user_id item_id`) based on playtime or review recommendation.
  - Between games (`item_id item_id`) if they have a high number of shared players.
- **Edge Weights**:
  - Playtime (`playtime_forever`) to measure user engagement. And recommendation count from reviews. These two terms are involved in measurement of users-games edge. We may use a logarithm transformation here since the difference between players' playtime may be quite big. The ownership of bundles for specific game will also increase the weight of the edge.
  - For the game-game edge, we may consider the **Jaccard similarity**. This measurement indicates the two games may be similar due to the high number of shared number. Also, the **Cosine similarity** based on players' time spent on the game may work here.

Purpose: This graph allows us to find similar users, recommend new games based on co-played titles, and analyze the most influential games in the network.

2. Tree (Game Category Hierarchy)

- **Nodes**:
  - Root: "All Games"
  - Intermediate nodes: Game genres (e.g., `RPG`, `Action`, `Indie`)
  - Leaf nodes: Specific games (`item_id`).

Purpose: Ensures genre-based filtering, allowing recommendations to align with the player's preferred category. It works like a filter.
Editted: To be honsest, the Tree part is more similar to the so-called "traditional recommender" in some way. Our group is seeking how to improve it, or we will just include two model in our program: Traditional search based on Tree and personalized search based on the graph.

3. Calculations in the plan
Note: The calculation methods are theoretical so we may not apply all of them in the project. Some of them are mentioned in part above. The calculation are mainly the weight of edges, they are important in the recommandation mechanism. Obviously, we need to rearrange the datasets and filtrate what we need before all of thoese.

For the Tree, we tempoarily have no ideas for the calculations. A possible idea is to measure the **Popularity of games** so we can change the sequence of games under different genres. The popularity is affected by the number of gamers, gamers' playing time and if the gamers recommend the game. Maybe we can use the feature of BST here.

In addition, we don't fully understand the requirement that: we need to design some of our **own algarithm** to handle most of the calculation. Sadly, we couldn't find how can we apply such in the project. **Please give some possible suggestions! Please !!!!**

Expected Display of the Program

Personalized model:
The user will start by inputting the games they have played. This can be done through a search bar or a clickable list of games. The system will retrieve relevant playtime data and user interactions if available.

The ideal mechanism:
- The input from the user of the program is `G_input`, which is the name of a game.
- Based on the weight of `user-game` edge, the program can find the players of `G_input`.
- Based on the weight of `game-game` edge, the program can conclude the similarity between `G_input` with some other games. Then we can find similar game `G_similar`.
- Then we can primarily generate a list of recommanded games.

We hope to have a 3D Graph Visualization (Upcoming Feature):
Users can explore the game recommendation network in a three-dimensional space, rotating and zooming in to see relationships between games more clearly.
A personalized list of recommended games is generated, including:
Game title, genre, and a brief description
Links to the game's store page
Recommendation reasoning (e.g., "Similar to [Game X]", "Popular among similar players"). This can be applied with some tags.

Additional Modules

We may additionally use, maybe we won't use some of them or choose alternative modules:

- pandas: Used for loading, filtering, and transforming JSON data into structured dataframes.
- numpy: Used for numerical computations
- scipy.spatial.distance and sklearn.metrics.pairwise: Used to compute Jaccard similarity and cosine similarity between games.
- networkx: Used for creating the bipartite user-game graph and performing graph algorithms
- plotly: Used for 3D network visualizations, allowing users to rotate and explore the game graph dynamically.
- streamlit: Used for building an interactive web-based UI for users to input their played games and see recommendations dynamically.

References

- McAuley, J. (n.d.). Datasets. UC San Diego, Computer Science and Engineering. Retrieved March 4, 2025, from https://cseweb.ucsd.edu/ jmcauley/datasets.html
- **Self-attentive sequential recommendation** Wang-Cheng Kang, Julian McAuley ICDM, 2018
- **Item recommendation on monotonic behavior chains** Mengting Wan, Julian McAuley RecSys, 2018
- **Generating and personalizing bundle recommendations on Steam** Apurva Pathak, Kshitiz Gupta, Julian McAuley SIGIR, 2017