Game Recommender System

Group Members: 20k-1050, 20k-0329, 20k-0375

# Introduction

With dozens of games available on platforms such as Steam, it can be difficult for players to choose which titles to play. In this project, we have created a game recommender system that employes multiple recommendation techniques that we learned in the course, on the *Steam* game datasets to recommend games to users.

# Dataset Details

We will be using steam games dataset comprising of:

* Steam\_games.csv
  + The "steam\_games.csv" dataset on Kaggle contains information about video games available on the Steam platform. Some of the attributes are name, release\_date, publisher, developer, popular\_tags, game\_description, genre, price, etc
* Steam-200k.csv
  + The "steam\_200k.csv" dataset available on Kaggle contains information about user reviews and game ownership on the Steam platform. The attributes are user\_id, game\_title, type(purchase/play), no.of hours played.

# Model Explanation

## Item-based Collaborative Filtering

In this technique, we have made a collaborative filtering recommendation system that uses the playtime data of users to predict game ratings. It employs the Pearson correlation coefficient to measure similarity between games and computes predictions based on weighted averages of ratings from similar games. Below are the main points in the method:

1. **Pearson Correlation Function:** The pearson\_correlation function calculates the Pearson correlation coefficient between two games based on user playtime. It determines the similarity between games by comparing the playtime data of common users.
2. **Predict Rating Function:** The predict\_rating function predicts the rating for a given user and game. It finds similar games to the user's played games based on Pearson correlation. The function calculates a weighted sum of ratings, where weights are the similarities between games, and predicts the rating by dividing the weighted sum by the total similarity.
3. **Prediction:** The code sets the user ID, game title, and the number of similar games to consider. It calls the predict\_rating function with these parameters and obtains the predicted rating.
4. **Output:** The code prints the predicted rating for the specified user and game.

## User-based Collaborative Filtering

This technique is designed to recommend games to a user based on their past gaming behavior using collaborative filtering. Here's an explanation of the code:

1. **Reading and Preprocessing the Data:**

The code starts by reading the 'steam-200k.csv' file using pd.read\_csv and assigning appropriate column names to the DataFrame.

It selects only the relevant columns: 'user\_id', 'game\_title', 'behavior', and 'hours'.

It further filters the data by removing entries where the number of hours played is less than or equal to 10.

1. **Creating a Pivot Table:**

The DataFrame is transformed into a pivot table using pd.pivot\_table. This operation reshapes the data so that each row represents a user, each column represents a game title, and the cell values represent the hours played.

1. **Computing Game Correlation:**

The correlation matrix is calculated using the corr() method on the pivot table. It computes the pairwise correlation between all pairs of games based on the users who have played them.

1. **Recommending Games:**

The recommend\_games function takes a user ID as input. It retrieves the games played by the user from the pivot table.

For each game played by the user, it finds similar games based on the correlation matrix and computes a weighted score for each similar game.

The weighted scores are summed up, and the recommendations are generated by excluding the games the user has already played.

The recommendations are sorted in descending order of the summed scores.

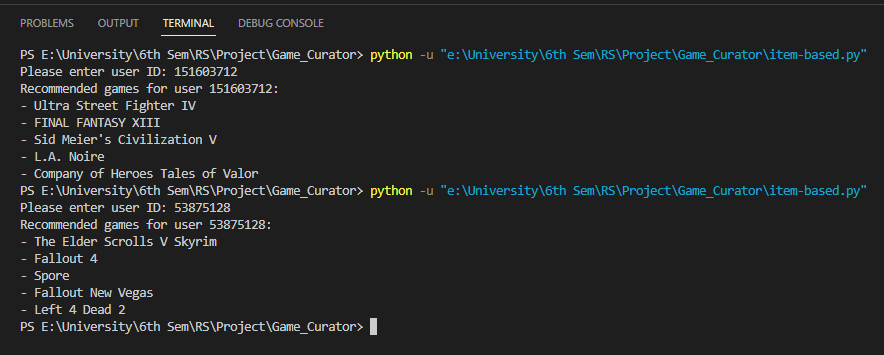
## Content-based Recommendations

In this technique, the dataset is loaded, and relevant features are selected. The data is preprocessed, combining the selected features into a single 'text' column. TF-IDF matrix was created to represent the textual content of the games. Cosine similarity is then computed based on the TF-IDF matrix.

The “get\_recommendations” function takes a game title as input and returns a list of recommended game titles based on cosine similarity scores. It retrieves the index of the input game title, sorts the games based on similarity scores, and selects the top recommended games. The recommended game titles are printed on the console.

# Experiments

## Item-based Collaborative Filtering



## User-based Collaborative Filtering

A picture containing text, screenshot, font, software

Description automatically generated

## Content-based Recommendations

A screen shot of a computer

Description automatically generated with low confidence

# Conclusion (Results)

## Content-based Recommendations

The code currently uses a subset of the dataset (10,000 rows) for efficiency reasons. Using a larger dataset could potentially lead to more diverse and accurate recommendations. The code assumes that the input game title exists in the dataset to find similar games. However, if the input game is new or has limited information, it might be challenging to generate accurate recommendations. The code currently does not consider user preferences or behavior. Incorporating user data, such as past interactions, ratings, or user profiles, can enable personalized recommendations tailored to individual preferences.

## Item-based Collaborative Filtering

The code currently provides general recommendations for a given user based on similarities with other users. To enhance the user experience, incorporating user-specific information (e.g., demographics, preferences, browsing history) can lead to personalized recommendations that better match individual tastes.

## User-based Collaborative Filtering

The code computes the correlation between each pair of users, which can be computationally expensive for large datasets. To make the approach scalable, techniques like matrix factorization, neighborhood-based methods, or dimensionality reduction can be explored. The dataset may suffer from data sparsity, where users have rated only a small fraction of the available games. This can limit the accuracy of the recommendations, especially for users with limited ratings.