SI507 Final Project Proposal

Jiajing Zhu - jiajingz

I. Motivation and Summary

Have you ever encountered a situation like this: After a hard day's work, you want to relax by playing video games, but you find it hard to find one that meets the criteria and you end up spending all the time searching and becoming more tired?

Currently, many platforms offer recommendations to help people find games they want to play, such as the top 100 rating games or the top 100 most popular games. However, sometimes these games are different from what people want. Based on the case, what we want to build is a search engine that ranks the most relevant games that meet the user's requirements. Users can submit game-related inquiries into our system, which will take into account the game's name, description, ratings, and other details, and will finally return a comprehensive list of the most pertinent games.

Through this system, we aim to improve the game user experience by making life easier for people who want to relax through games without putting much effort into the annoying search process.

II. Database

A. Data Source

In this project, I mainly use two data sources, one is fetched by crawling the web pages, and the other one is an existing dataset obtained from Kaggle.

1. Steam (https://store.steampowered.com/). I plan to get the detailed information of a variety of games to form our database, including:

- Title: the name of the game
- Price: the sale price
- Genre: the genre list of the game
- Description: the game introduction
- Developer: the developer list of the game
- ReviewSummary: the overall favorable rate

2. Steam Store Games (Clean dataset) from Kaggle (https://www.kaggle.com/datasets/nikdavis/steam-store-games). There are several datasets we will use from this category. They serve as a supplement of the data crawled from Steam. Initially, we plan to use the data in steam.csv and steam_requirements_data.csv to supplement our database, using information including:

- ReleaseDate: the release date of the game
- Platforms: the supported platforms
- RequiredAge: the required age of the game

- PCRequirement: the computer configuration requirement for PC
- MacRequirement: the computer configuration requirement for Mac
- LinuxRequirement: the computer configuration requirement for Linux

B. Data Collection

We apply the BeautifulSoup library to pull the data from the Steam database (https://store.steampowered.com/). We first extract the link and the ID of the games and save it as a local temporary file. Then, according to each link, we use requests.get(link) and then bs4 to extract keywords, including names, prices and types.

As for the steam store games, we download the dataset from the website (https://www.kaggle.com/datasets/nikdavis/steam-store-games).

III. Methods

A. Node Creation:

1. **Games**: Represent each game as a node. Attributes can include Title, Price, Description, Developer, Review Summary, Release Date, Platforms, Required Age, PC/Mac/Linux Requirements.

2. **Genres**: Represent each genre as a node.

3. **Developers**: Represent each developer as a node.

4. **Platforms**: Represent each platform as a node (e.g., Windows, Mac, Linux).

B. Edge Creation:

1. **Game-Genre Edges**: Connect games to their genres.

2. **Game-Developer Edges**: Connect games to their developers.

3. **Game-Platform Edges**: Connect games to the platforms they are available on.

C. Graph Construction:

Use a library like `networkx` in Python to construct the graph.

IV. Data Interaction and Visualization

I will assume the data is presented through a web interface, and we will use libraries like `D3.js` for front-end visualizations and `networkx` for server-side graph operations.

A. Game Node Interaction:

When a user selects a game node:

1. **Genre Distribution**: Show a pie chart of the distribution of genres of games similar to the selected game (based on shared developers, genres, or platforms).

2. **Developer Connection**: Display a subgraph showing how this game is connected to other games through shared developers.

3. **Platform Availability**: Show a bar chart displaying the availability of similar games across different platforms.

4. **Price Distribution**: Show a histogram of the price distribution of similar games.

B. Genre Node Interaction:

When a user selects a genre node:

1. **Popular Games**: List the top N popular games in this genre based on review summary or other criteria.

2. **Genre Evolution Over Time**: Display a line graph showing the number of games in this genre released each year.

3. **Price Trends**: Show a line graph of average game prices in this genre over time.

4. **Developer Collaboration**: Display a network graph showing developers that have frequently released games in this genre.

C. Developer Node Interaction:

When a user selects a developer node:

1. **Game Portfolio**: Display all the games developed by this developer, along with key information.

2. **Genre Expertise**: Show a pie chart of the genre distribution of games developed by this developer.

3. **Collaboration Network**: Display a network graph showing collaboration with other developers, if any.

4. **Review Summary Distribution**: Show a bar chart of the review summary distribution of games developed by this developer.

D. Platform Node Interaction:

When a user selects a platform node:

1. **Game Availability**: List all the games available on this platform.
2. **Popular Genres**: Show a bar chart of the most popular genres on this platform.
3. **Average Price**: Display the average price of games on this platform.
4. **Age Distribution**: Show a histogram of the required age of games on this platform.

V. Backend to Frontend Communication

I can set up API endpoints using Flask to serve the necessary data to the front end for visualization. The front end sends a request to the backend specifying the node selected and the type of information required, and the backend processes the graph to return the relevant data.