Name: Andrew To

ID: 14591152 / dt686

DSCI-351

Project Report

1. Project Description

Title: Steam Video Game Recommender System

This project aims to build a recommender system for Steam using combination of user review, user & item data, item metadata that can be successfully and effectively implemented for Steam's recommender systems.

2. Data Description

- **a. Source:** https://cseweb.ucsd.edu/~jmcauley/datasets.html#steam_data
- **b. Description: 3 main datasets: (I will only mention main features here)**
  - i. df1:
    1. 25799 samples (~60k samples after exploding reviews column)
    2. Features:
       - a. user_id
       - b. reviews
         - **i.** item_id
         - **ii.** posted
         - **iii.** recommended (Used for validation)
  - ii. df2:
    1. 88310 samples (~4M samples after exploding reviews column)
    2. Features:
       - a. user_id
       - b. item_id
       - c. item_name
       - d. playtime_forever
       - e. playtime_2weeks
    3. The playtime data in this dataset is converted to custom ratings of a user for a game using log
  - iii. Df4:
    1. Item metadata: Product features
    2. Features:
       - a. publisher
       - b. genres
       - c. title
       - d. tags
       - e. specs

3. Data featuring

a. Training/Testing Dataset: df2 + df4:

i. Combine genres, tags, specs, publisher, developer to create a soup, and then use tf-idf vectorizer for the soup to extract the information from metadata

ii. Use playtime_forever and playtime_2weeks to make a custom "ratings" using log to determine how much interest an user has for a game

b. Validation Dataset: df1

i. df1 has the column "recommend", which is a Boolean column. I use rows where the value in "recommend" column is True to extract information and compare with the metadata of the top-k recommendation to evaluate the performance

ii. Merge df1+df2+df4 together to get games that appear in all 3 datasets

4. Model used and results

- I utilize the content-based model, which is very helpful for datasets which have much item-side information. With this large amount of item metadata and users' behavior with these items, we can create a user profile from item cosine similarity combined with implicit rating created from the total time and user spend for a game

- 2 methods to find recommended games for an user:

    - Item-item only: Assume that the most played game is the strongest sample and use item cosine similarity only to find most related games
    - Utilize item cosine similarity and implicit rating from playtime data: Create a rt_matrix of target_user across all items (including games with 0 minutes played). Compute scores as the dot product of this rt_matrix with the cosine_similarity (from metadata tf-idf vectorizer) to select the top-rated games

5. Evaluation

- Here, I use the word "recommended game" for the game that an user clicked recommend (shown in df1), and "predicted game" for the game that is suggested for this user
- I'm not using cosine similarity between games for evaluation since I believe it can cause misunderstanding (if the cosine similarity is poorly calculated, the predicted games could have good cosine similarity with recommended games, but not really "similar")

- Instead, I'm using the following 2 ways, using truly recommended games from df1
    - Do not exclude played games when making prediction (such that users may see games they played in the suggestion list), then check whether the games that they liked (recommended) are in the suggestion list
    - Exclude played games when making prediction to avoid overfitting, then use the following rule with metadata to evaluate:
        - Across 3 metadata fields - genres, tags, specs – calculate the percentage of features that appear more than five times within the set of 10 predicted games
        - Example: genres: 100%; tags: 50%; specs: 50%
        - These percentages show how close is the recommended game is to the set of predicted games

6. Limitation & Future improvement

a. Limitation:

i. The data after exploding fields is so large, so it takes time for merge/preprocessing/computation

ii. A lot of missing values, so I couldn't use everything in the data source

iii. Skip some important features, especially review text since we have limited data for this. There is another dataset which has a lot of review text but it doesn't have user id (it only has username) so I can't merge

b. Future plan:

i. Use some methods of data crawler to collect user_id for its username, so that we can optimize the text review dataset

ii. Try with knowledge-based RS with bundle data