Steam Game Recommender

Create recommender system to suggest games to users

Domain background

Steam is a gaming platform where multiple games on different platforms (including PC, Mac, and Linux). It also has a social component including streaming and community forums. Many video game providers also hold frequent discounts on the offered games. Users can also see recommendations based on the games they played already.

Improving the recommendations shown to a user can increase user engagement and increase revenue of the game producer (and Steam too). Given the huge variety of games available across various platforms, targeting the correct user who is likely to buy the game, conduct micro-transactions, and/or recommend the game to his friends is a very lucrative problem for game producers. Since Steam receives a cut on games sold through Steam, Valve also has an interest in properly recommending games.

Past Research

There have been previous models based on the playtime data (described below). [1] creates a top-L recommender using 2 different models - 1) A factor based model which imputes the missing playtime values and 2) A nearest neighbor model.

While [2] doesn't create a recommender model, player behavior was analysed across 4 experiments - 1) Playtime distribution - players : The majority of players are dedicated to just one or a few games 2) Playtime distribution - games : There are 4 major types of games depending on how their playtimes are distributed. 3) Game ownership patterns : Seeing which games occur together in a player's library 4)Ranks/Reviews vs Ownership/Playtime : Review scores for a game are not correlated to sales and playtimes.

Kevin Wong also created a recommender system using 2 different recommendation algorithms: 1)Pearson R correlation and 2) Log-Likelihood ratio

In this capstone, a recommender system is built to suggest games for the users.

Datasets

Steam Video Games dataset

URL : https://www.kaggle.com/tamber/steam-video-games

This dataset provided by Tamber. It contains user data for different Steam games, namely the hours a particular user has played a particular game.

To quote the content description from the Kaggle page - “This dataset is a list of user behaviors, with columns: user-id, game-title, behavior-name, value. The behaviors included are 'purchase' and 'play'. The value indicates the degree to which the behavior was performed - in the case of 'purchase' the value is always 1, and in the case of 'play' the value represents the number of hours the user has played the game.”

DBPedia

URL : http://wiki.dbpedia.org/about

To quote the official website, “DBpedia is a crowd-sourced community effort to extract structured information from Wikipedia and make this information available on the Web.” I’ll be using DBPedia to add genre and Series information to the games in the main Steam dataset.
Example - http://dbpedia.org/page/Fallout_4
Game - Fallout_4
Series - Fallout_(series)
Genre - Action_role-playing_game

Scoring Function

A scoring function will be defined to incorporate the wide range in hours played across different users and different games. This will be an attempt to reduce skew and normalise the given data.

Score = Personal Ratio * Normalised Hours

Example
If user XYZ has played game A for 20 hours and he has totally played 100 hours on Steam, his personal ratio is 20/100 = 0.2.
But what if game A is a singleplayer campaign which only lasts around 10 hours? Then user XYZ seems to like game A a lot. It is assumed that the players playing game A are normally distributed. If game A is distributed with a mean of 10 hours and a standard deviation of 5 hours, XYZ’s normalised hours is (20-10)/5 = 2.
Therefore, user XYZ’s score for game A is 2 * 0.2 = **0.4**

Problem statement

Create a recommendation system which can predict user engagement with games (the scoring function which is defined below) given playtime data about the games and the user. Top-5 games will be recommended to the user along with the predicted score (defined above). This score can be used by companies to decide revenues. Eg. If a user plays a game (eg Candy Crush) very much as compared to other players, similar long-term games with micro-transactions can be suggested to him. On the other hand, if a user plays a lot of different games but doesn’t spend a lot of time on any of them, another variety of games with one-time prices can be shown to him.

Solution

Collaborative filtering will be used to fill in the missing user-game score matrix. The top 5 will then be chosen and shown to the user as recommendations. Matrix factorization will be used to characterise the latent features between users and games. Since game data is also available (from DBPedia) factorisation machine models will also be used to improve performance.

Since explicit data is available (in the form of the defined scores), a factorization_recommender will be used as suggested here. Note : item_similarity_recommender will not be used since it can not incorporate user and game specific information (as noted in the relevant documentation Notes section). Parameter tuning will then be done to improve performance.

Benchmark model

Since the dataset is likely to be very sparse, a base popularity_recommender will be used as the benchmark model. This will help determine whether the augmented model performs better or worse than the simplest model. This will also judge whether the extra space and computation required for the augmented model is worth any improvements in accuracy.

Evaluation metrics

Instead of using “Precision at N” as used by Kevin Wong, an RMSE(Root Mean Squared Error) score will be used since the predicted score is more important in this case than just predicting whether the game is played or not. Eg. Suggesting “Candy Crush” to a player who plays such micro-transaction games for a long time will give more revenue than suggesting a game which has a one time price.
A portion of the user-game data will be used as the test set using random_split_by_user. This will then be used to decide the final performance of the model.

Project Design

1. Download Steam dataset
2. Add DBPedia information to games
    a. Decide how many genres to keep (Top 5? Top 10?)
3. Split data into training-test
    a. Consider splitting data into a cross validation set too (Depending on the amount of data available)
4. Create scoring function
   NOTE: Do not use the test data while creating the scoring function
5. Create benchmark model using popularity_recommender.
   Both the training data and the CV data will be used for this step
6. Create base facorization_recommender model
7. Hyperparameter tuning using GridSearch on the CV data
   Required hyperparameters - num_factors, regularization, linear_regularization, max_iterations and sgd_step_size
8. Decide final model and compare model performance on test dataset