Project 3: Hybrid Game Recommendation System: Combining Collaborative Filtering and Content-Based Method

Dirk Wouter Bester
Principles of Data Science - CS771

October 24, 2024

1 Introduction

Recommender systems have become essential in many industries to enhance user experiences and drive engagement. These systems leverage historical user data to provide personalized suggestions, such as movie recommendations on Netflix, product recommendations on Amazon, or content suggestions on YouTube. The gaming world is exponentially expanding with 3.32 billion active gamers worldwide, thus the need for a good recommender is urgent. Gaming is also something I engage with and that is why I chose this topic. So our goal is to create a personalised game recommender. I chose 2 datasets for this project to create a **Hybrid Recommendation System**.

Steam Games Dataset (Steam Games Dataset, MIT via Steam Spy, Kaggle): This dataset was created at MIT using Steam Spy. It contains information about 96500 games on the steam database. It is a very extensive dataset with 39 collumns of data such as game names, peek concurrent users, genres, tags and many more. I will leverage this dataset for my **Content-based Recommendation System**.

Game Recommendations On Steam (Game Recommendations on Steam Kaggle): This dataset contains over 41 million cleaned and preprocessed user recommendations (reviews) from a Steam Store. It has User reviews with product ID to user ID relations, which indicates how many time users spent on different games. I will leverage this dataset for my **Collaborative Filtering System**.

The cold start problem could easily be solved by boosting the most popular games as this ensures a wide variety of genres and titles would be recommended. The final recommender yields good results with combined tecniques.

2 Recommender System Process

The type of recommender system implemented for this project is a **Hybrid Recommendation System**. I decided to combine the use of **Collaborative Filtering** and **Content-based Filtering**. This ensures that by combining the techniques, the recommender system will recommend games not only based on the similarity of genres and tags but also on previous user interaction. The important thing here is to weigh th two correctly, put too much emphasis on the content-based filtering and you have no user-interaction to bring in variety of genres and vice versa.

2.1 User Based Collaborative Filtering

Collaborative filtering is a common method of personalized recommender system which filters information such as interactions data from other similar users. Since it works by predicting user ratings, it is considered as performing regression task. We will be using an user to user collaborative filtering. It basically operates under the assumption that users who gave similar ratings to a certain item are likely to have the same preference for other items as well.

Our **Supervised Learning model** makes use of **Singular Value Decomposition (SVD)**. We first encode our user and game id's to convert them to numerical form. We then identified hours as our target variable. After that we determine our traint/test split and we decide on a maximum training split to ensure our model gets the maximum amount of data to train with and our focus is maily on new users that is not in the dataset. We then make use of the Singular Value Decomposition (SVD) which is a matrix factorization method. It decomposes the user-game interaction matrix into latent factors that represent the underlying relationships between users and games. We decided on 10 epochs as we felt that this was sufficient and accurate enough. Lastly we train the model and save its files to our root directory.

This model then receives the play history of the user and the model searches for users that has played the same games. It then predicts the ratings for the other games and we apply a boost to the 5 games played most regularly alongside the input games.

2.2 Content-Based Filtering

Content-Based Filtering is when similarities are calculated over product metadata (features of a product), and it provides the opportunity to develop recommendations. The products that are most similar to the relevant product are recommended.

We started of by choosing the features that we will be using for the filtering. We decided to make use of genres, tags and categories. We also decided to repeat tags 3 times and categories 2 times to make these more important as they are more relevant for recommendations.

Now we can start with calculating the scores. First we have to extract the textual features using TF-IDF. **TF-IDF** is a technique to represent the importance of a word within a specific text relative to all other text in the dataset. It helps to capture meaningful words while ignoring common words that appear frequently across many documents.

Now we apply this TF-IDF on the combined text data that gives us vectors. We can now calculate the similarity scores. There are 2 options: Euclidean Distance and Cosine Similarity.

We decided on using cosine similarity as it is independent on the magnitude of the text and some games have much more genres and tags than others. **Cosine Similarity** is the measure of similarity between two non-zero vectors widely applied in many machine learning and data analysis applications. It measures the cosine of the angle between two vectors.

Now because game names are much shorter than the other data we decided to brake the names up into n-grams. Now instead of usign Cosine Similarity for analyzing the n-grams we started using **Jaccard Similarity**. The Jaccard similarity is well-suited for comparing sets of items, which in this case are the n-grams derived from each game title.

3 Implementation

The implementation consisted of the following key steps:

3.1 Data Cleaning

Both the datasets had to be cleaned.

The **Steam Games Dataset** contained way to many games that was not played by anyone. So the first thing I did was filter out the top 15% on peak concurrent users to make my database much more relevant and efficient to work with. This left me with just over 1000 games remaining. I also decided to just keep the 24 columns relevant to what I wanted to do.

The **Game Recommendations On Steam** is a massive dataset with over 41 million entries. 41 Million is non realistic to work with when you want to do supervised learning so this also had to be reduced. I also had to change it to clean using smaller chunk sizes otherwise my VsCode would crash. So I removed all the entries where the hours played were less than 250 and the games that was not recommended. This made it a much more viable to work with. I also just kept the 5 important columns: name, is_recommended, user_id, app_id and hours played. I also reduced it a bit more by removing the games that was linked with just one game because this would ultimately be useless for my goal in mind.

3.2 Content-Based Filtering

The Content-Based filtering is then done. Now we have to make weight for how much the similarity will contribute and then how much the title similarity would contribute and we decided on 60% for the normal similarity and 40% for the title similarity. We also decided to make use of weights for how long a user has played their input games. For example if an user played Rust for 1000 hours and Chess for 1 hour it would heavily favour the recommendations on Rust as the user probably enjoyed it more.

Now to account for the **Cold Start Problem**. Since you would have no information on the user when it is cold start you have to use pre-existing data in the dataset. We decided to make use of a popularity boost. that boosts the games score based on how popular is via the peak concurrent users variable. This ensure that that the first recommendations tends to be the most played games. This also ensures a wide variety of genres.

3.3 User-Based Collaborative Filtering

The collaborative filtering model was implemented using the **Surprise** library with SVD. After the model was ran it returns the 5 games with the highest predicted ratings (hours) and then a boost of up to 20% is applied.

3.4 Custom GUI Interface

I also decided to create a custom GUI using customTkinter. This would make the initial input of age, language and budget much more user-friendly and would also provide a efficient way to input play history.

It starts off with three windows that asks for initial parameters like age, budget and language. This immediately then filters out the games not in those constraints. You then reach a window where you have to input the games you have played and for how long you played them. For a cold start you could just click through and enter no games.

The GUI then loads and waits for the recommenders to do their jobs. It then displays the top 10 games by recommendation score, it also displays an image by fetching there header images online. Also when clicked on the image the website of that specific game loads. If you want to see more games just scroll down and press see more.

3.5 Hybrid System Breakdown

The hybrid recommendation system is broken down as follows:

- Similarity Score: 60
- Title Similarity Score: 40
- Collaborative Score: 20 (boost)
- Popularity Score: 15 (boost)

These weights was carefully chosen and fine-tuned to achieve the best results.

3.6 Possible Improvements

An enhancement I considered is using NLP with BERT to analyze the "About this Game" descriptions. This would provide a deeper understanding of each game's content, allowing for more precise matching and adding another parameter to fine-tune the recommendations. This could further improve the system's accuracy and relevance.

4 Results

4.1 Cold Start Problem

Here are some screenshots of the final recommendation GUI when running a cold start as well as the results:

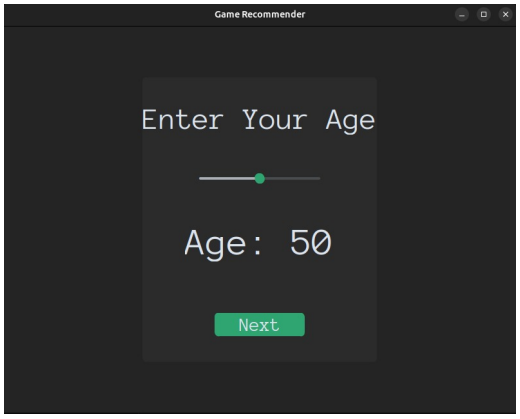

Figure 1: GUI Window 1

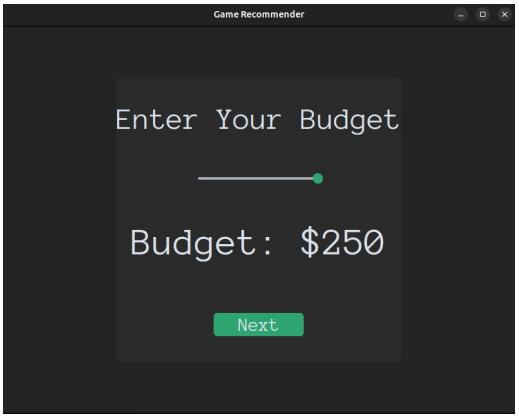

Figure 2: GUI Window 2

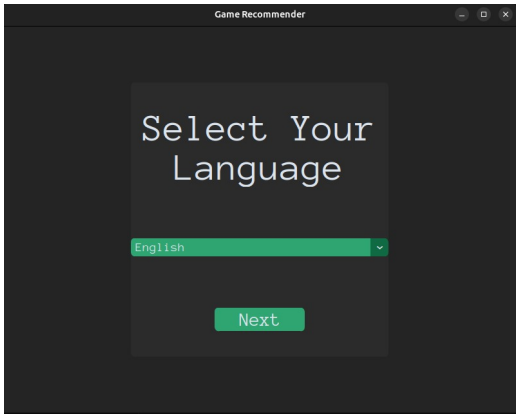

Figure 3: GUI Window 3

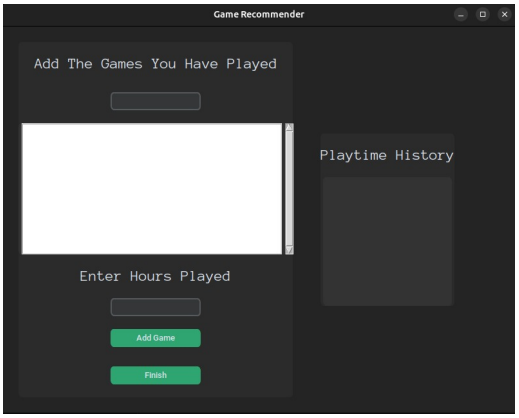

Figure 4: GUI Window 4

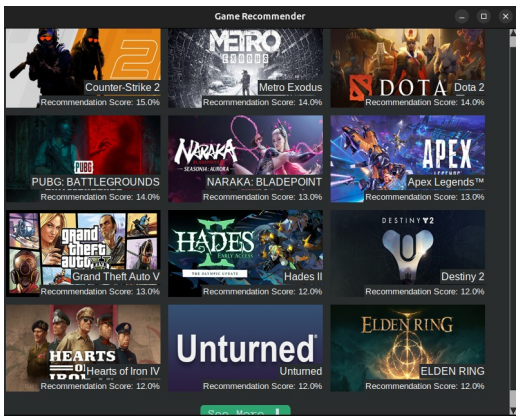

Figure 5: Cold Start Results - 1

```
Game: Counter-Strike 2
 - Recommendation Score: 15.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 15.00%
----------------------------------------
Game: Metro Exodus
 - Recommendation Score: 14.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 14.44%
----------------------------------------
Game: Dota 2
 - Recommendation Score: 14.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 14.24%
----------------------------------------
Game: PUBG: BATTLEGROUNDS
 - Recommendation Score: 14.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 14.11%
----------------------------------------
Game: NARAKA: BLADEPOINT
 - Recommendation Score: 13.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 13.28%
----------------------------------------
Game: Apex Legends™
 - Recommendation Score: 13.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 12.79%
----------------------------------------
Game: Grand Theft Auto V
 - Recommendation Score: 13.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 0.00%
 - Popularity Contribution: 12.53%
----------------------------------------
```

Figure 6: Cold Start Results - 2

As you can see from the cold start results the games that are more popular, which automatically means its more fun for the general population are put on top and the popularity is the only thing that is considered in this case. This also means that the results gotten have a wide variety of genres and gives the new user much more to choose from. Games like CS-2 is very good starting point for new gamers and games like GTA-V is open-world so the user are free to do what they like.

4.2 Collaborative Score Results

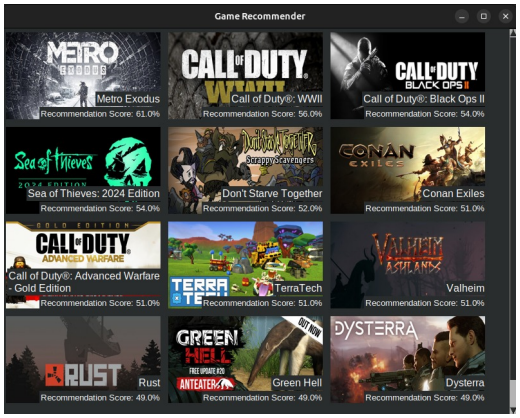

Figure 7: Recommendations For Terraria

```
----------------------------------------
Game: Sea of Thieves: 2024 Edition
 - Recommendation Score: 54.0
 - Collaborative Contribution: 9.00%
 - Similarity Contribution: 34.85%
 - Popularity Contribution: 10.10%
----------------------------------------
```

Figure 8: Collaborative Score Boost

The game input as play history was Terraria. When you go look manually in the recommendations.csv dataset you can look up the user ID's that play terraria and can see the two games are played together regularly. On the right you an see the Collaborative contribution of 9% that was boosted because our model picked this up.

4.3 Similarity Score Results (Content-Based)

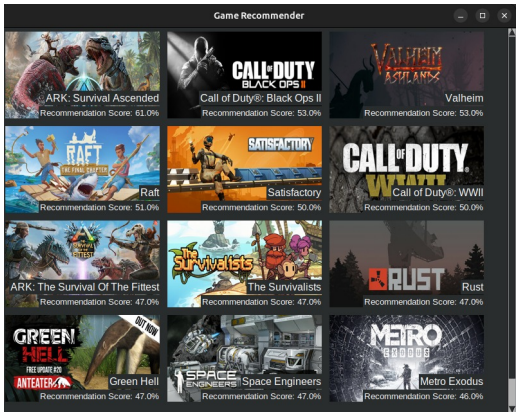

Figure 9: Recommendations for ARK: Survival Evolved

```
Game: ARK: Survival Ascended
 - Recommendation Score: 61.0
 - Collaborative Contribution: 0.00%
 - Similarity Contribution: 51.00%
 - Popularity Contribution: 10.48%
----------------------------------------
```

Figure 10: Similarity Score Contribution

The game input as play history was ARK: Survival Evolved, just to see how good our similarity score calculator was. It was inevitable that the other two ARK games made this list because of the title similarity as well as it being games that are very much the same. Then you can also see games like raft which is basically the same game as ARK it is just animals trying to kill you instead of dinosaurs. Then Green Hell is also a very good recommendation because it also has the same exact concept as ARK. So we can say with confidence that the similarity score is pretty accurate at identifying similar games based of the textual features.

4.4 Game Hours: Weight System

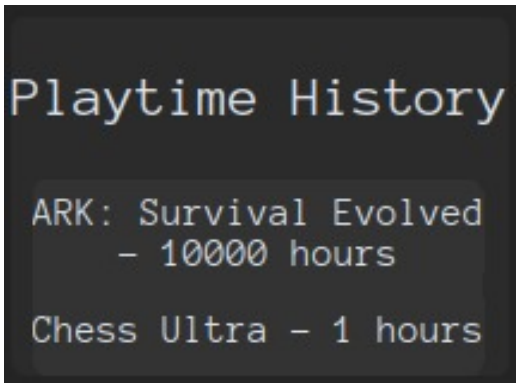

Figure 11: Uneven Weights

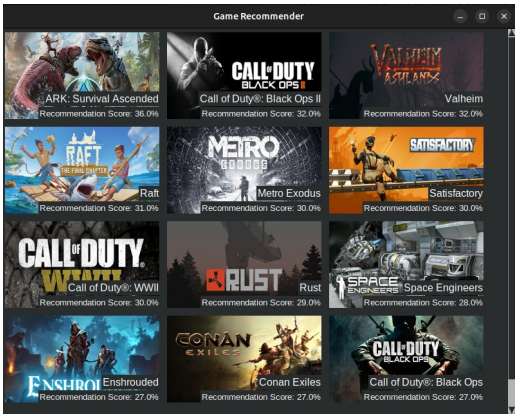

Figure 12: ARK: 1000 hours, Chess: 1 hour

Here we input 2 games but with very different weights. We input ARK again with 10000 hours to show how similar it is to previous results and Chess 1 hour. The recommendations show that there was not very much change as the user probably enjoyed playing ARK way more as they put in 9999 hours more.

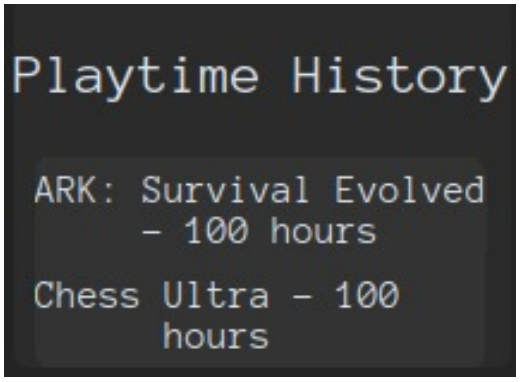

Figure 13: Even Weights

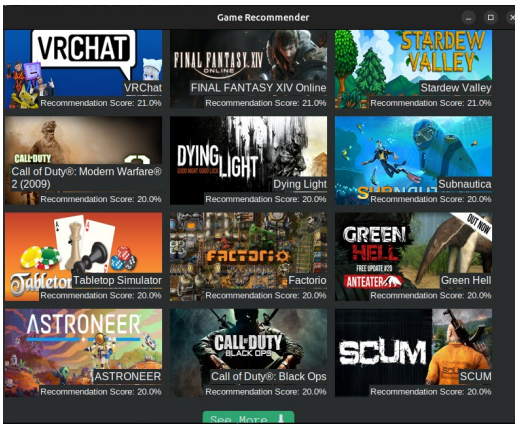

Figure 14: ARK: 100 hours, Chess: 100 hours

Here we again input 2 games with the same weight to showcase the recommendations is then a good mix between the two games. The games recommended is now a combination of both genres and this is what we expected.

4.5 Conclusion

In conclusion, this project successfully implemented a Hybrid Recommendation System that combines both Collaborative Filtering and Content-Based Filtering to recommend games based on user play history and game metadata. In my opinion the results looks very good and recommends very well by combining the two techniques very well. The filtering at the start also ensures the user does not receive recommendations outside his/her constraints. I actually let my friends play around with this recommender and they were quite astonished as it recommended games they have also played with their play history, which also makes me think this was a very successful project.

5 References

- Surprise Python Library for Collaborative Filtering.
- TF-IDF in Scikit-learn.
- Steam Games Dataset – MIT via Steam Spy, Kaggle.
- Game Recommendations on Steam – Anton Kozyriev, Kaggle.
- Exploding Topics: Global Gamers Statistics.
- Google Developers: Collaborative Filtering Basics.
- Medium: Collaborative Filtering in Recommender Systems.
- Medium: Content-Based Filtering in Recommender Systems.
- GeeksforGeeks: Cosine Similarity Explained.