Collaborative Filtering for Steam Games Recommendation

William Evan Lomanto
Computer Science Department
Bina Nusantara University
Jakarta, Indonesia
william.lomanto@binus.ac.id

Verry Andrian
Computer Science Department
Bina Nusantara University
Jakarta, Indonesia
verry.andrian@binus.ac.id

Said Achmad
Computer Science Department
Bina Nusantara University
Jakarta, Indonesia
said.achmad@binus.ac.id

Rhio Sutoyo
Computer Science Department
Bina Nusantara University
Jakarta, Indonesia
rhio.sutoyo@binus.ac.id

2023 5th International Conference on Cybernetics and Intelligent System (ICORIS) | 979-8-3503-6948-9/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICORIS60118.2023.10352243

***Abstract*—This research explores collaborative filtering with Singular Value Decomposition (SVD) and Pearson correlation to provide game recommendations on the Steam digital distribution service dataset. Collaborative Filtering aims to leverage user preferences to identify similar gaming patterns. This can be done through SVD and Pearson correlation algorithms. SVD reduces the dimensionality of the game rating matrix, while Pearson correlation measures the similarity between users and games. The proposed method generates personalized recommendations based on user preferences and opinions of similar gamers. Evaluation metrics include Mean Average Error (MAE) and Root Mean Square Deviation (RMSE). Results demonstrate the effectiveness of CF with SVD and Pearson correlation in delivering relevant and personalized game suggestions on Steam based dataset.**

Keywords*—*Collaborative Filtering, Steam Games, Recommendation, Singular Value Decomposition (SVD), Pearson Correlation

I. Introduction

Video games are a typical recreational activity that provides a sensory experience akin to other recreational activities, such as listening to music. By extension, gamers are driven by a desire for novel encounters, similar to a musicophile who loves listening to new music. However, pursuing novelty does not always work in the gamer's favor. For instance, if a user favors adventure games, they may not be interested in a limited selection of games in the role-playing games (RPGs) genre even though the games have an excellent rating [1]. There seems to be a sweet spot where a game can make gamers enjoy the novelty of a new experience, but it is familiar enough that gamers do not become discouraged from trying it.

From a developer's standpoint, crafting a successful and widely embraced game poses a formidable challenge. Distribution can be a considerable challenge. Independent distribution can have a substantial financial cost while not guaranteeing your game is seen by the right audience. Therefore, platform distribution has become more commonplace. Steam1 digital distribution service emerged as the original platform, initially serving as a software client for disseminating digital games. Over time, Steam gained popularity through social media, streamers, communities, and various other channels. Nevertheless, the vast number of games available on Steam oftenleaves users struggling to discover titles that align with their preferences.

[1]https://store.steampowered.com/

Within the Steam platform, users can review games, ex- pressing both positive and negative opinions. These reviews are crucial in determining whether a game provides players with a favorable or unfavorable experience. This paper aims to leverage these reviews as a tool for aiding in the development of a recommendation system [2]. By employing collaborative recommendation systems on Steam, personalized game sug- gestions can be offered to users based on their past behavior and preferences. To generate tailored recommendations, these systems analyze various data points, including the user's play history, purchase history, and reviews. Utilizing a collaborative recommendation system enables users to explore new games they may not have encountered. For instance, if a user prefers strategy games, the recommendation system may propose other games that align with their interests. Among the various recommendation systems available, this paper focuses on user- based collaborative filtering, a type of Collaborative filtering. What makes collaborative filtering different from the other recommendation methods is the use of comparative mea- surement between users' preferences. In this context, user preferences can be indicated explicitly in the form of the user's submitted ratings or implicitly in the document of monitoring user's behavior like their purchase history [3]. The collaborative filtering method, however, requires a database of users' preferences to be already available. This can be a problem when a service using a collaborative filtering method is in its early stage, thus only having data to a limited set of users' preferences. This problem makes early users experience unsatisfactory recommendation results or poor performance in general. This performance issue will improve as the database used for collaborative filtering increases in numbers and diversity.

Collaborative filtering can harness accessible data, such as implicit preferences derived from a user's rating list. Ad- ditionally, the communities of games and users within the system can serve as additional data sources to enhance the accuracy of the collaborative filtering algorithm. For instance, this paper [4] utilizes historical purchases, game time, and the relationship between items to build a collaborative filtering game-recommender model.

The algorithm can identify games frequently played in conjunction or with similar game-play mechanics, enabling it to recommend such games to users who enjoy any of the games within the associated group. By leveraging the entire implicit data accumulated from the preceding state- ments, collaborative filtering algorithms can discern patterns and similarities that may not be immediately evident, thus facilitating the generation of more precise recommendations customized to each user's unique preferences.

In brief, collaborative filtering is an approach that leverages historical data from multiple users to provide item recom- mendations to other users [5]. One research paper proposes a recommendation system that employs collaborative filtering based on implicit preferences derived from ratings. The dataset from the Steam platform was filtered based on game playing times, followed by Python to categorize the game times and examine seven recommendation algorithms. The findings indicate that, within the dataset utilized, no notable distinction is observed among the explored algorithms [6].

This paper aims to employ the fundamental principles of collaborative filtering techniques to recommend Steam games to users. This involves developing a program capable of gener- ating output that aligns with the preferences of individual users based on the gaming choices of other users with similar tastes on a technical and theoretical level. This provides insights into the performance of different CF methods in a new environment different from the commonly used benchmark of Movielens or Netflix database.

II. RELATED WORK

Collaborative filtering is a generally known topic for a recommendation system. It is usually used in a rating-based user feedback system like a movie or a book, as seen in Esmael Ahmed's article [7]. Joon Seok Lee stated that no one most effective way to use a collaborative system exists. The result may vary depending on the data set and problem parameter [8]. Collaborative filtering also has a few inherent long-standing problems, like the cold start problem. Cold start problems occur when the database supporting a collaborative filtering algorithm doesn't have enough data to keep an accurate recommendation. Many varied solutions have been proposed, tackling many different aspects of the problem. Some of these solutions include mean normalization and using a secondary algorithm to cover for collaborative filtering weaknesses. Mean normalization of the rating can mitigate the cold start problem.

and make the process more efficient. Zhihui Zhou proposed to jump-start the cold start into a warm start state by integrating content-based generative models into his collaborative filtering algorithm [9]. Other than the cold start problem, there are many ways to make collaborative filtering more practical, like pairing Pearson Correlation with Firefly algorithm [10]. Many studies have been conducted to improve the efficiency and effectiveness of collaborative filtering. Some tried to improve the efficiency by combining collaborative filtering with another algorithm more suited to the purpose of the recommendation system. This can be observed in Venkata Bhanu and Evani Venkateswara's approach, implementing some elements from content-based filtering into collaborative filtering to create hybrid collaborative filtering more suited for an e-learning platform [11].

A. *Type of Collaborative Filtering*

Collaborative filtering can be categorized into three groups. These include memory-based, model-based, and hybrid recom- menders, each with strengths and weaknesses. Memory-based collaborative filtering, like Pearson Correlation, is easily im- plementable but has problems with scalability in larger datasets and data sparsity. Model-based collaborative filtering like SVD has improved prediction performance and data sparsity prob- lem management but takes more resources to build and can lose useful information when implementing dimensionality reduction techniques. Hybrid collaborative filtering is often the better of the three with its ability to complement the weakness of one method with another method. However, it is highly complex and often requires external information that's usually not publicly available [12].

B. *Equations for Evaluations*

The traditional constrained Pearson Correlation coefficient and SVD will be used as the base lgorithm for this project, although a newer algorithm has been proposed [13]. Pearson correlation (1) measures the linear correlation between two variables. In the context of collaborative recommendation, it quantifies the similarity between two users or two items using their rating pattern, i.e., the user's rating compared to the mean rating. The SVD model that will be used is FunkSVD (2), the basic idea of it can be described as given a scoring matrix R = (rmn) ∗ r ∗ l = [r1, r2, ..., rl], the model will find an approximate matrix A to approximate M to predict the missing score [14].

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2} \qquad (1)$$

$$min_{u,v} \sum_{(i,j)\in R} \left(r_{ij} - \hat{r}ij\right)^2 + \lambda \left(\sum i|u_i|^2 + \sum_j |v_j|^2\right) \qquad (2)$$

The mean absolute error (3) and root mean square error (4) will be used for the evaluation method. The value of both methods will be utilized and compared to determine the success of our collaborative filtering algorithm [15]. pairing Pearson Correlation with Firefly algorithm [10].

$$MAE = \frac{1}{n}\sum\nolimits_{i=1}^{n}|y_i - \hat{y}_i| \qquad (3)$$

$$RMSE = \sqrt{\frac{1}{n}\sum\nolimits_{i=1}^{n}(y_i - \hat{y}_i)^2} \qquad (4)$$

III. METHODOLOGY

The system architecture of this research is shown in Algo- rithm 1, and the data for this experiment is obtained through Kaggle.com [16]. The data contained 12,393 users and their interactions with 185,673 games. The details of the unique value of the data used can be seen in Table I. Users interact with the games library by purchasing and playing the game. The playing time of each game is tracked by the hours. The hours played become the primary variable of which the recommendation of a game is pushed since a high play hour is generally connected to a favorable experience within the game.

Algorithm 1 Recommendation System

```
1:  Select users with "played" game status
2:  Convert hour_ played into simulated_rating
3:  if SVD then
4:      Load data into Surprise dataset
5:      Perform train-test split
6:      Initialize SVD model and fit it on training_ set
7:      Predict ratings for test_ set
8:  else
9:      if Pearson Correlation then
10:         Create simulated_rating matrix
11:         Normalize the converted matrix
12:         Create similarity_weight matrix between users
13:     Create similarity games matrix
14:     end if
15: end if
16: Calculate each game's score
17: Push the top recommendation
```

TABLE I. THE UNIQUE VALUE OF DATAFRAMES USED

Data Type	Unique Value
user id	12393
played games	5155
hourplayed	1593

A. *Selecting user who has "played" status on their game*

The data is filtered only to display users who have played the game by only selecting users whose hour-played exceeds

1.0 and have "played" status. The dataset automatically sets the hour played to 1.0 if the game is purchased. Thus, the system filters out users who only bought the game but did not play it afterward by only looking at users with "played" status.

B. *Converting the hours played into simulated rating*

Since the data has no explicit rating, the algorithm needs other data elements to replace the user rating. Thus, the hour played is used as the primary metric. We must gather the average hours played for each game to convert hours played into a rating that can be used in collaborative filtering. Each user's playtime hour will then be compared to the average playtime hour of the general user for that game. If the player exceeds the average playtime hour, their rating will be automatically set to five. If it is below the average playtime hour, the system will divide it by the average playtime hour and multiply it by five to count "how much percentage of five stars" your rating is. The proposed formula is shown in (5).

$$\text{Simulated Rating} = \frac{hour\ played}{average\ hourplayed}\ x\ 5 \qquad (5)$$

C. *SVD*

1) Load data into Surprise dataset:

The Surprise[2] dataset is a library for creating recommender systems. In this step, a data frame composed of user id, game name, and converted rating will be loaded into the Surprise dataset alongside a rating scale set from 1 to 5. After loading the data frame, it creates a Surprise dataset object that will be used later to utilize Surprise library functions.

2) Perform train-test split:

Using the SVD library, the Surprise dataset is split into a training and testing subset. In this paper, the split is 20% into training and 80% into testing. A random state parameter is also set up to guarantee that the exact split will be obtained if the code is executed multiple times with the same parameter values.

3) Initialize SVD model and fit on trainset:

This step trains the SVD model on the previously formed training subset. The training is done through matrix factorization and analyzing the latent factor of a dataset. This process is called fitting and is achieved through Surprise's "fit" method. This fitting process is done to make the SVD model learn the latent factor of the dataset, capturing the pattern that enables it to predict the user's rating.

4) Predict ratings for testset

The SVD model trained on the training subset is used to predict the user rating on the testing subset. This process uses the "test" method available in Surprise's library. The output prediction can later be compared to the actual rating in the testing subset to be evaluated using methods such as MAE and RMSE.

D. *Pearson Correlation*

1) Make converted rating matrix

The top 100 games with the "played" status ratings will be gathered and turned into a matrix to find similarity weight. The top 100 selection is made to increase efficiency due to Google collab's limitation on data processing[3]. This process is done to make applying the similarity weight of a user to his/her rating easier down the line.

2) Normalize the converted matrix

The converted matrix is normalized to help address potential biases and inconsistencies in the ratings of the dataset. This leads to a better and more accurate similarity weight measurement.

3) Make similarity weight matrix between user

The algorithm will find the similarity weight using the Pearson Correlation technique shown in Equation (1), which is the base of the collaborative filtering algorithm.

Similarity weight is a value that determines how similar the current user is compared to other users. The system will pick games that the current user has played and apply the Pearson Correlation formula to many users to get the similarity weight between our current user and every other user.

[2]https://surpriselib.com/

[3]https://research.google.com/colaboratory/faq.html

That value will represent the current user's proximity to others in the data set. Once the similarity weight is found, it will be multiplied by the converted rating in our matrix, producing a weighted rating matrix. Weighted rating will represent the opinion of the current user's neighbors regarding each item that has the potential to be recommended.

4) Make similarity games matrix

The system will make a data frame matrix comprising the previous normalized matrix and the user's similarity weight matrix. The data frame will select the rows from the normalized matrix where the row index is present in the index of the user's similarity weight matrix, which contains similar user ids. As such, the similarity games matrix will contain only the similar user ids and the games they have rated with its user-item pair similarity value.

E. Calculate game score and push top recommendation

1) SVD

Using the user id and list of unique games provided in the input, the SVD model will estimate the rating of the provided user id to every game in the list of unique games based on the pattern learned through analyzing the training subset given earlier. Using the "predict" method in Surprises, these user-item estimations can be outputted into an object form.

2) Pearson Correlation

The predicted game rating is put into a game score array. The game score is calculated through the user-item similarity value available in the similarity games matrix times the user-user similarity value found in the sim- ilarity weight matrix between users. The predicted score will be arranged into a data frame sorted in ascending order.

IV. Result and Discussion

A. Dataframes used

The primary data frame is structured in four columns in user_id, game, status, and hour played as seen in Tables II. The 'purchased' status entry is discarded, leaving only played games with each user_id's hour played. The second data frame, called "mean," is made to collect the top 100 games' average hour play between users. These data frames are then merged into the main data frame to calculate the simulated rating and make the rating matrix normalized.

Table III shows the data after processing, indicating that the data frame has a lot of empty data fields. This is because the proportion of users to games is heavily skewed towards the number of games. Users generally rate only a fraction of the entire game dataset. It is unrealistic for users to rate and interact with more than ten games in this 30,000-game data frame. This creates a data frame with a high sparsity between each relevant data.

TABLE II. Sample Table of Raw Data Used

Game	User ID	Hour Played	Mean Hours Played	Rating
The Elder Scrolls V Skyrim	151603712	273.0	104.71	5.00
	59945701	58.0	104.71	2.77
	92107940	110.0	104.71	5.00
	250006052	465.0	104.71	5.00
	11373749	220.0	104.71	5.00
Deathmatch Classic	58298928	0.2	0.74	1.35
	66748534	0.2	0.74	1.35
	19094181	0.4	0.74	2.70
	2110581	0.3	0.74	2.03
	71871620	0.4	0.74	2.70

TABLE III. Visualization of Data Sparsity After Processed

User ID	Game Ratings			
	APB Reloaded	AdVenture Capitalist	Age of Empires II HD	Alien Swarm
5250	NaN	NaN	NaN	4.70
76767	NaN	NaN	1.91	0.77
86540	NaN	NaN	0.10	NaN
144736	NaN	NaN	NaN	NaN
181212	NaN	NaN	NaN	NaN

B. Method Performances

This study uses two methods to produce a recommendation list, i.e., Pearson Correlation (PC) and SVD. The summary is shown in Fig. 1. The system is tested through the same input of user id and evaluated in MAE and RMSE. Regarding their differences, RMSE is more sensitive to deviation than MAE. It means that if the recommendation prediction has an outlier, the RMSE result will jump higher than the MAE. Moreover, both methods produce different results.

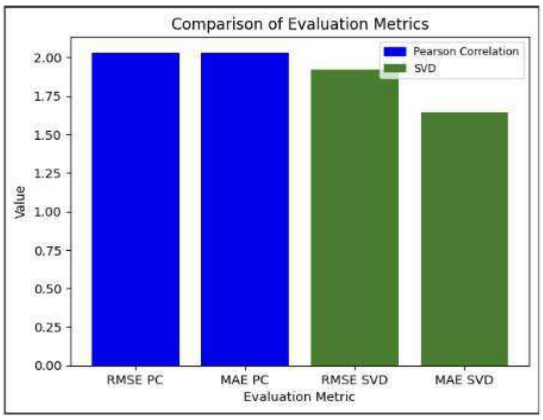
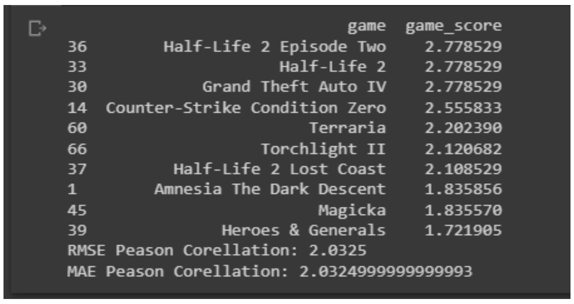


Fig. 1. Comparison of Evaluation Metrics

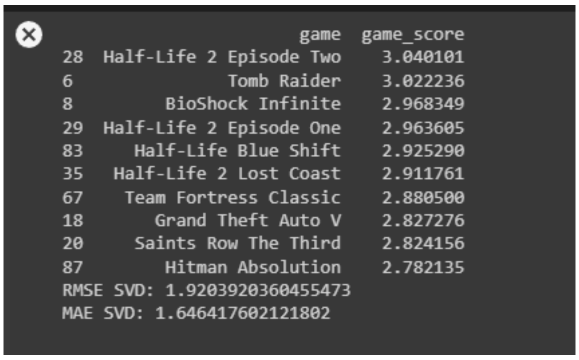
When MAE and RMSE evaluation is executed, SVD performs better in both metrics than Pearson Correlation. This is related to memory-based CF weaknesses in a database that shows extreme sparsity. In this database, Pearson correlation struggles to correlate and find the nearest neighbor of a user since the database lacks overlapping value, resulting in an unreliable neighborhood. An unreliable neighborhood will result in unfocused and inaccurate recommendations reflected in Pearson correlation being outperformed in its RMSE and MAE score. Figure 2 shows the output with Pearson.



```
                          game  game_score
36        Half-Life 2 Episode Two    2.778529
33                Half-Life 2    2.778529
30          Grand Theft Auto IV    2.778529
14  Counter-Strike Condition Zero    2.555833
60                    Terraria    2.202390
66                Torchlight II    2.120682
37        Half-Life 2 Lost Coast    2.108529
1         Amnesia The Dark Descent    1.835856
45                     Magicka    1.835570
39           Heroes & Generals    1.721905
RMSE Peason Corellation: 2.0325
MAE Peason Corellation: 2.0324999999999993
```

Fig. 2. Pearson Correlation Output

SVD outperforms Pearson Correlation by a sizeable margin (above 0.1) in RMSE. However, there is a staggering difference in MAE. Indicating that SVD is outperforming Pearson correlation in both metrics. Since SVD is a model-based CF, it has more ways to combat the extreme data sparsity present in this database, such as using dimensionality reduction techniques such as LSA (Latent Semantic Analysis). Through dimensionality reduction, the SVD algorithm can target relevant users or items more effectively and capture transitive relationships between users faster than Pearson correlation. A trend can be seen continued from previous works [17] in the value of its RMSE and MAE as SVD typically outperformed Pearson correlation and many memory-based CF in a high sparsity database. Figure 3 shows the output with SVD.



```
                     game  game_score
28  Half-Life 2 Episode Two    3.040101
6               Tomb Raider    3.022236
8          BioShock Infinite    2.968349
29  Half-Life 2 Episode One    2.963605
83       Half-Life Blue Shift    2.925290
35    Half-Life 2 Lost Coast    2.911761
67     Team Fortress Classic    2.880500
18          Grand Theft Auto V    2.827276
20       Saints Row The Third    2.824156
87         Hitman Absolution    2.782135
RMSE SVD: 1.9203920360455473
MAE SVD: 1.646417602121802
```

Fig. 3. SVD Output

The result shows the strengths and weaknesses of Pearson correlations and SVD. In comparison, SVD is outperforming Pearson correlation in both fields. Since RMSE is more sensitive to deviation than MAE, it can be observed that there are fewer outliers in the Pearson correlation method. SVD is expected to perform well in this type of database because of its scale and data sparsity. However, the consistency of the Pearson correlation doesn't rule out the possibility of it being used in a database more suited for it.

V. Conclusion and Future Works

By implementing both recommendation algorithm in a simple recommendation system, it can be seen that SVD outperform Pearson correlation. When given a large dataset of items (30k) with data sparsity, the Pearson correlation method of comparing each user one by one makes it inefficient and less accurate. On the other hand, SVD can handle scalability issues and data sparsity better than Pearson correlation, making it better than Pearson correlation in a dataset with extensive user-item interaction like Steam. SVD is expected to perform well in this type of database because of its scale and data sparsity. However, the consistency of the Pearson correlation does not rule out the possibility of it being used in a database more suited for it.

Future work can use hybrid recommenders to make a collaborative recommendation system that is more tailor-made for the Steam dataset. Such methods can be something like Hybrid Matrix Factorization(HMF) that combines memory- based and model-based CF, applying a memory-based col- laborative filtering method to identify a subset of similar users or items, reducing the search for relevant neighbors and then using matrix factorization to enable efficient and accurate recommendations of those subsets. Other than further optimization, future works can also handle the practicality of the collaborative filtering method, testing its theoretical performances and user satisfaction performance.

References

[1] G. Cheuque, J. Guzma´n, and D. Parra, "Recommender systems for online video game platforms: The case of steam," in *Companion Proceedings of The 2019 World Wide Web Conference*, 2019, pp. 763– 771.

[2] D. Lin, C.-P. Bezemer, Y. Zou, and A. E. Hassan, "An empirical study of game reviews on the steam platform," *Empirical Software Engineering*, vol. 24, pp. 170–207, 2019.

[3] M. Grcˇar, D. Mladenicˇ, B. Fortuna, and M. Grobelnik, "Data sparsity issues in the collaborative filtering framework," in *Advances in Web Min- ing and Web Usage Analysis: 7th International Workshop on Knowledge Discovery on the Web, WebKDD 2005, Chicago, IL, USA, August 21, 2005. Revised Papers 7*. Springer, 2006, pp. 58–76.

[4] Q. Li, X. Liang, C. Su, and Y. Wang, "A game recommendation method based on machine learning," in *2021 IEEE 3rd International Conference on Frontiers Technology of Information and Computer (ICFTIC)*, 2021, pp. 307–312.

[5] R. Bunga, F. Batista, and R. Ribeiro, "From implicit preferences to ratings: video games recommendation based on collaborative filtering," *From implicit preferences to ratings: Video games recommendation based on collaborative filtering*, pp. 209–216, 2021.

[6] A. Girsang, B. Al Faruq, H. Herlianto, and S. Simbolon, "Collaborative recommendation system in users of anime films," in *Journal of Physics: Conference Series*, vol. 1566, no. 1. IOP Publishing, 2020, p. 012057.

[7] E. Ahmed, A. Letta *et al.*, "Book recommendation using collaborative filtering algorithm," *Applied Computational Intelligence and Soft Com- puting*, vol. 2023, 2023.

[8] J. Lee, M. Sun, and G. Lebanon, "A comparative study of collaborative filtering algorithms," *arXiv preprint arXiv:1205.3193*, 2012.

[9] Z. Zhou, L. Zhang, and N. Yang, "Contrastive collaborative filtering for cold-start item recommendation," *arXiv preprint arXiv:2302.02151*, 2023.

[10] M. M. Dewi, "Optimasi pearson correlation untuk sistem rekomendasi menggunakan algoritma firefly," *Jurnal Informatika*, vol. 9, no. 1, pp. 1–5, 2022.

[11] V. B. P. Tolety and E. V. Prasad, "Hybrid content and collaborative filter- ing based recommendation system for e-learning platforms," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1543–1549, 2022.

[12] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[13] V. B. P. Tolety and E. V. Prasad, "Hybrid content and collaborative filter- ing based recommendation system for e-learning platforms," *Bulletin of Electrical Engineering and Informatics*, vol. 11, no. 3, pp. 1543–1549, 2022.

[14] Y. Xiaochen and L. Qicheng, “Parallel algorithm of improved funksvd based on gpu,” *IEEE Access*, vol. 10, pp. 26 002–26 010, 2022.

[15] T. O. Hodson, “Root-mean-square error (rmse) or mean absolute error (mae): when to use them or not,” *Geoscientific Model Development*, vol. 15, no. 14, pp. 5481–5487, 2022.

[16] “Steam Video Games,” May 2023, [Online; accessed 16. May 2023]. [Online]. Available: https://www.kaggle.com/datasets/tamber/steam-video-games

[17] X. Su and T. M. Khoshgoftaar, “A survey of collaborative filtering techniques,” *Advances in artificial intelligence*, vol. 2009, 2009.