Game Recommendation System Using Steam Game Dataset - Final Report

Ioannis Kalamakis
Department of Electrical Computer Engineer
University of Thessaly
Volos, Greece
ikalamakis@uth.gr

Tryfon Gkelos
Department of Electrical Computer Engineer
University of Thessaly
Volos, Greece
tgkelos@uth.gr

Abstract

In recent years, the gaming industry has experienced unprecedented growth, resulting in a market flooded with a vast array of game titles across multiple genres. This abundance has posed a significant challenge for users in discovering games that align with their unique preferences. Our project addresses this challenge by developing an intelligent Game Recommendation System. The system is designed to offer personalized game suggestions, thereby enhancing user satisfaction and engagement in the expansive gaming world.

Our recommendation system leverages the comprehensive Steam Game Dataset, which includes a wide range of data such as user reviews, game genres, release dates, and player engagement metrics. This diverse dataset allows us to extract valuable insights into user preferences and game popularity, which are crucial for building an effective recommendation system. Our approach combines collaborative filtering, matrix factorization techniques, and Singular Value Decomposition (SVD) to identify patterns in user-item interactions and generate tailored recommendations.

Preliminary experiments conducted using this dataset have shown promising results in terms of recommendation accuracy and user satisfaction. By comparing our system with a baseline model based on popularity-based recommendations, we have established a benchmark for our approach. As the gaming landscape continues to evolve, our system is designed to adapt and update recommendations, ensuring a seamless and personalized gaming experience for users.

1 Introduction

In today's era, the gaming industry has experienced rapid and substantial growth into a vast sea of digital entertainment, offering a wide array of games across various genres. This plethora of options has given rise to a new challenge: individuals seeking leisure and a moment of relaxation through gaming are now often overwhelmed by the sheer volume of choices. The quest to find the perfect

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

game that aligns with one's preferences can unintentionally consume the time meant for relaxation and escape from daily stresses.

1.1 Problem Statement

The importance of this issue parallels the need for ease in leisure activities—just as some find solace in reading a book, watching television, or enjoying a drink, others seek the same comfort in the immersive worlds of video games. Thus, the problem extends beyond mere inconvenience; it is about preserving quality time in a world filled with digital excess.

1.2 How It Works

To address this, we have developed a recommendation system where the input is a user's unique ID. Using advanced techniques such as Gradient Descent, Singular Value Decomposition (SVD), and neural networks, our algorithm outputs a personalized list of game recommendations.

2 Literature Review

The landscape of game recommendation systems features diverse methodologies, each addressing the challenge of user preference prediction in unique ways. This review examines key studies and categorizes them into distinct approaches, discussing their strengths, weaknesses, and how they compare to our work.

2.1 Neural Collaborative Filtering

In "Neural Collaborative Filtering vs. Matrix Factorization Revisited," the focus is on enhancing collaborative filtering with deep learning techniques. The strength of this approach lies in its ability to capture complex user-item interactions, but it may require extensive computational resources. Our work benefits from this approach by integrating neural networks for more nuanced recommendations.

2.2 Deep Learning in Recommendation Systems

"Use of Deep Learning in Modern Recommendation System: A Summary of Recent Works" offers a comprehensive overview of deep learning applications in recommendation systems. The paper underscores the versatility and effectiveness of deep learning in capturing intricate patterns in data, aligning with our neural network-centric methodology.

2.3 Collaborative Filtering with Game Theory

The paper "A New Collaborative Filtering Approach Based on Game Theory for Recommendation Systems" introduces game theory to collaborative filtering. This approach provides a unique perspective on user interactions, though it may not fully address content-based aspects, which our system incorporates.

2.4 Content-Based Filtering for Personalized Gamification

"Recommender Systems for Personalized Gamification" explores content-based filtering in gaming. This approach excels in offering highly personalized recommendations based on game characteristics, similar to our content-based filtering component.

2.5 Hybrid Recommendation Models

In "A Survey of Recommendation Systems: Recommendation Models Techniques and Application Fields," hybrid recommendation models combining collaborative and content-based methods are discussed. This aligns closely with our approach, which also leverages the strengths of both methods for a comprehensive recommendation system.

In conclusion, we find that hybrid models and those incorporating neural networks represent the state-of-the-art in game recommendation systems. Our work echoes these advancements by combining

collaborative filtering, content-based filtering, and neural network techniques for a robust and adaptive recommendation system.

3 Dataset and Features

3.1 Dataset Description

Our project employs two datasets from the Steam platform: ‘steam games‘ and ‘steam-200k‘. The ‘steam games‘ dataset consists of 40,833 entries, encompassing a broad range of data including game URLs, types, names, descriptions, reviews, release dates, and more. The ‘steam-200k‘ dataset comprises 199,999 entries, detailing user-game interactions such as user IDs, game titles, and hours played.

3.2 Preprocessing and Feature Extraction

The initial dataset, referred to as the user dataset, encompasses various data points, including the user's ID, the game title, the user's activity ('purchase' or 'play'), and a corresponding value linked to that activity. Each entry in this dataset represents a user's interaction with a game, which can either be 'play' or 'purchase'. When the activity is 'play', the associated value represents the number of hours played. Conversely, when the activity is 'purchase', the associated value is consistently set to 1, indicating that the user bought the game.

To prepare the dataset for analysis, we undertake preprocessing and feature extraction steps. Specifically, we have reorganized the 'behavior' column into two separate columns: 'purchase' and 'play'. In this new format, each row contains a 'play' value of 1 if the user engaged in gameplay or 0 if there is no recorded playtime.

This preprocessing and feature extraction are crucial to facilitate our subsequent analysis of the data and to extract meaningful insights from the user dataset.

The second dataset, known as the game dataset, comprises a comprehensive list of games along with various associated attributes. This dataset includes information such as game descriptions, direct Steam store URLs, package types (e.g., app, bundle), game titles, concise descriptions, recent reviews, total reviews, release dates, developers, publishers, popular tags (such as Gore, Action, Shooter, PvP), game details (like Multi-player, Single-player, Full controller support), supported languages, achievements, genres (including Action, Adventure, RPG, Strategy), detailed game descriptions, mature content descriptions, minimum system requirements for game execution, recommended system requirements, original prices, and discounted prices. In total, the dataset encompasses a vast collection of 51,920 distinct games.

3.3 Feature Engineering and Neural Network

In our approach to enhancing the recommendation system, we deviate from traditional preprocessing methods by adopting a more nuanced and neural network-oriented strategy. Unlike the initial dataset preparation that segregates 'play' and 'purchase' activities into separate columns, our preprocessing focuses on retaining only 'play' activities. This choice is guided by our aim to analyze user engagement with games based on playtime, which is more indicative of user preferences than mere purchase history.

In the user dataset, we transform the user's ID and game name columns into categorical codes, a step not explicitly mentioned in the traditional approach. This transformation is crucial for our neural network model, as it facilitates the handling of categorical data and enables the model to efficiently process and learn from these features.

Furthermore, we introduce a normalization step using MinMaxScaler for the 'value' column, representing the duration of gameplay. This normalization is essential to ensure that the input features are on a comparable scale, enhancing the neural network's ability to learn effectively from the data.

Another significant departure from the described traditional approach is the exclusion of the 'purchase' data. We focus exclusively on the 'play' behavior, as it provides a more accurate reflection of user preferences and engagement with games. This decision aligns with our objective to tailor recommendations based on actual user-game interactions, rather than mere acquisition of games.

Our preprocessing and feature engineering steps are specifically designed to align with the pecific features of neural network models, enabling our system to uncover deeper insights and patterns in user-game interactions. This approach not only streamlines the data for neural network processing but also ensures that the recommendations generated are more personalized and relevant to each user's unique gaming preferences.

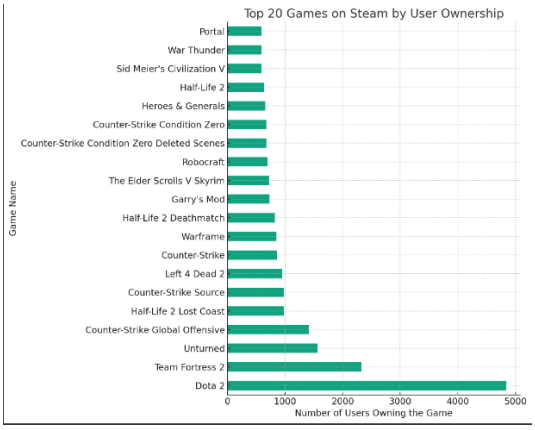
3.4 Dataset Citation

For this project, two distinct datasets are utilized, both sourced from Kaggle and containing data extracted from Steam, a popular digital distribution platform for video games.

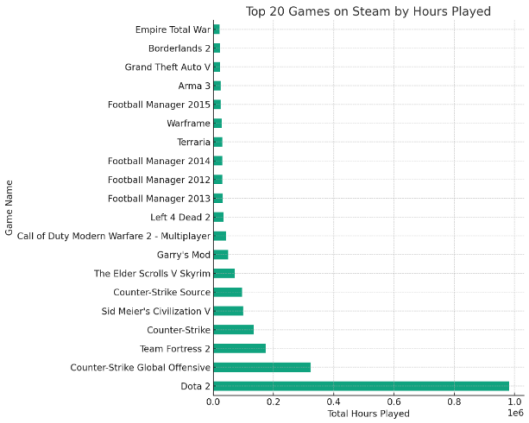
3.5 Examples and Visualization

The following visualizations offer insights into the gaming preferences and behaviors of Steam users based on the aforementioned datasets.

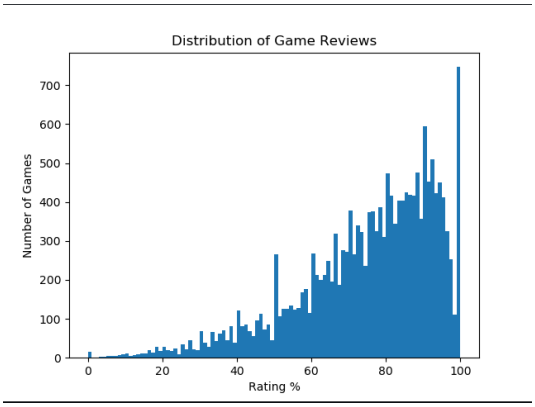
In the first plot, we observe the top 20 games on Steam, ranked by user ownership. This metric reflects the number of users who have purchased each game.



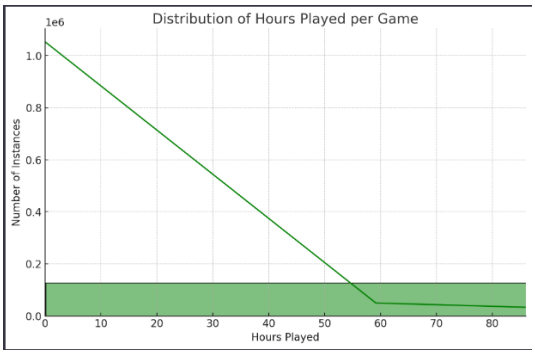
The second graph presents a surprising revelation: the most played games in terms of total hours do not necessarily correlate with the games that have the highest ownership. This indicates that certain games, despite not being as widely owned, engage players for longer durations.



Another intriguing aspect of the dataset is the overall rating of games on Steam. By analyzing the average rating of all games, we find that the majority of games on Steam receive an average rating above 65



Lastly, an analysis of the hours played per game reveals that most users do not spend more than 50 hours playing any single game. This information can be valuable for understanding user engagement and the longevity of interest in specific titles.



4 Methods

4.1 Overview

Our Game Recommendation System incorporates several techniques, content-based filtering, neural networks, and Expectation-Maximization (EM) algorithms. We have utilized these methods to analyze user-game interactions and predict user preferences effectively.

4.2 Train and Test Datasets

In preparation for the implementation of algorithms for the collaborative filtering recommender system, we create distinct training and testing datasets using the reformatted user dataset.

Our modified user dataset contains a total of 128,804 rows, each containing unique information pertaining to user-game interactions. To establish our train and test datasets, we opt to extract 20 percent of all user-game interactions, resulting in 25,761 rows designated for the test dataset, while the remaining 103,043 rows constitute the training dataset.

The training dataset serves as the foundation for constructing the collaborative filtering recommender models. After model development, these models are then applied to generate recommendations for all users listed within the test dataset.

4.3 Content-Based Filtering

The content-based recommender system operates by generating recommendations based on the similarity between games a user already possesses and other games in the dataset.

To build this recommender system, a series of steps are taken. Initially, we must prepare the data and develop the necessary algorithm. The process begins with the preprocessing of the game dataset. This dataset is formatted in a more streamlined manner to serve as input for the recommender algorithm. Additionally, percentages from game reviews are extracted, as they are an essential feature.

The recommender system's core functionality involves suggesting games similar to those already owned by a user. To prepare the data for this content-based recommender, we first select relevant information that is likely to be valuable for finding similar games. These selected columns are then extracted from the game dataset.

A critical decision is made to include only games that exist in both the game dataset and the user dataset. This choice is motivated by the fact that there are many games in the game dataset that have never been played or purchased by any user in the user dataset, rendering them irrelevant for the recommender system. Moreover, the size of the game dataset is prohibitive for creating a cosine similarity matrix due to its memory requirements.

To match games from both datasets, unique IDs are generated for each game by removing non-alphanumeric symbols, spaces, and converting all letters to lowercase. This ID transformation is also applied to games in the user dataset. Subsequently, all unique IDs from the user dataset are used to filter rows in the game dataset, resulting in a subset of 3,036 games that match some of the 5,152 games from the user dataset. This approach significantly improves the recommender system's performance compared to using only game titles, which initially resulted in only 71 matching games.

With the reduced game dataset, spaces in the selected columns are removed to prevent false matches. For instance, 'Assassin's Creed' is altered to 'AssasinsCreed' ensuring accurate matching. Custom columns are also created by combining information from multiple columns, allowing for the exploration of different recommendation strategies.

Further manipulation of the review column from the game dataset involves extracting the percentage and relevant qualitative review information. The recommender system relies on a cosine similarity matrix generated from the frequency matrix of words in the selected column. This matrix facilitates the calculation of similarity scores between games.

Recommendations for each user are obtained by combining recommendations produced for individual games. These recommendations are based on similarity scores from the cosine similarity matrix and are sorted by reviews, from the highest to lowest. If the list of recommendations is empty or invalid,

it is handled accordingly. Ultimately, the recommender system generates a CSV file containing user-specific recommendations based on their owned games and their corresponding reviews.

4.4 Neural Network

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the human brain. It is a type of machine learning process, called deep learning, that uses interconnected nodes or neurons in a layered structure that resembles the human brain. It creates an adaptive system that computers use to learn from their mistakes and improve continuously. Thus, artificial neural networks attempt to solve complicated problems, like summarizing documents or recognizing faces, with greater accuracy.

Embedding: The term "Embedding" in the context of machine learning and particularly in neural networks refers to a representation of discrete data in a continuous, high-dimensional space. This concept is termed "Embedding" because it involves embedding discrete, often categorical data (like words, user IDs, or item IDs) into a continuous vector space. Each point in this space represents an instance of the discrete data, and the positioning of these points captures significant semantic or contextual relationships.

Deep Learning Architecture: Our model's core is a deep neural network consisting of:

- **Input Layer:**
  - User input: An integer for user ID.
  - Game input: An integer for game ID.
- **Embedding Layers:**
  - User embedding: Maps user input to a dense vector.
  - Game embedding: Maps game input to a dense vector.
- **Merge Layer:** Concatenates user and game vectors.
- **Dense Layers:** Fully connected layers for learning non-linear feature combinations.
- **Output Layer:** A neuron predicting the user's game preference score.

Training and Optimization: The model is trained on user-game interactions, aiming to minimize the Mean Squared Error (MSE) loss function, defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of samples, y_i the actual score, and $\hat{y}_i$ the predicted score.

Optimizer: We use the Adam optimizer to update network weights based on loss function gradients. Adam combines AdaGrad and RMSProp's advantages, enhancing stochastic gradient descent's efficiency.

Model Customization and Performance Expectations

While developing our neural network-based recommendation system, we adhered to certain guidelines regarding the model architecture. However, it was necessary to make specific alterations to ensure compatibility with our dataset. These modifications, while essential for the model to compile and process our data effectively, may have impacted the overall performance of the system.

The customizations to the model architecture were driven by the characteristics and requirements of our dataset. Adapting the model to these specifics meant deviating from the ideal neural network architecture typically recommended for such tasks. As a result, the performance of our system, although satisfactory, might not fully reflect the potential of a more standard architecture.

It is important to note that achieving the optimal architecture and fine-tuning a neural network to suit specific data needs often goes beyond the scope of standard guidelines and expectations. In our case, the adjustments were a necessary compromise to align the model with our data, acknowledging that the ideal results may be attainable with further refinements and learnings beyond the scope of this class.

These considerations are crucial for understanding the context of our model's performance and the trade-offs involved in customizing neural network architectures for specific datasets.

4.5 Expectation-Maximization (EM) Algorithm

The Expectation-Maximization (EM) algorithm is a powerful tool for maximum likelihood estimation, especially in scenarios involving latent variables. It offers an effective approach for estimating the parameters of a given data distribution.

To devise a robust rating system, especially given the implicit nature of the user dataset, we opted to leverage the EM algorithm for modeling the distributions of hours played for each game. This choice was made in favor of using percentiles as an alternative.

Our rating system is constructed by analyzing the distribution of hours played for each game found in the user dataset. We categorized this data into five groups, which correspond to a 5-star rating system. This approach allows us to assign ratings that reflect a user's perception based on the hours they've invested in a game relative to other users.

It's worth noting that on the Steam platform, users have the option to request refunds for games with less than 2 hours of gameplay. We've taken this into account in our recommender system by excluding user-item interactions with less than 2 hours of gameplay.

To facilitate our recommender system, we create a user-item matrix, where users are represented as rows and games as columns. Any missing values in this matrix are set to zero. The values stored in the matrix represent the logarithm of hours played for each user-game combination. Following best practices, we limit our data to games with more than 50 users and users who have played a game for more than 2 hours.

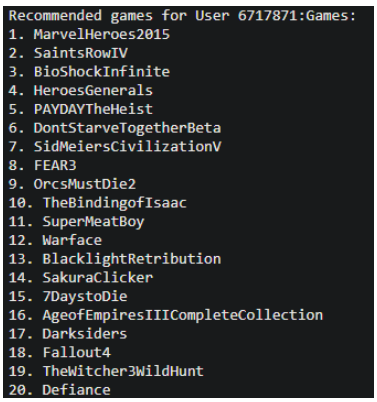
Initially, a basic Singular Value Decomposition (SVD) algorithm implementation proved inadequate for generating satisfactory recommendations from our dataset. Consequently, we opted for an implementation of the SVD algorithm utilizing a Gradient Descent approach..

This approach allows us to factorize the user-item matrix into singular vectors and singular values, resembling eigendecomposition. The Gradient Descent method helps address missing data by making predictions. It is a convex optimization technique that assists in finding optimal U and V matrices to represent the original user-item matrix. This replacement involves estimating new values for missing entries based on information from similar users and games.

We set the learning rate to 0.001 and the number of iterations to 200. Throughout this process, we monitor the Root Mean Square Error (RMSE) to evaluate the algorithm's performance. The U and V matrices are initialized with random values drawn from a normal distribution within the range [0, 0.01].

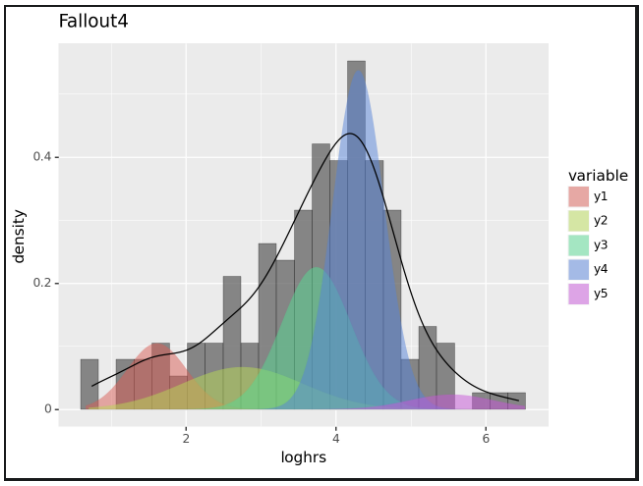
The RMSE is calculated to measure the disparity between actual values and predicted values, providing insights into the effectiveness of our recommender system.

Also we made sure that when the user of our user interface inputs a user that does not exist in the CSV file , the code gives him the top 20 games based on hours played.
Here is an output for a random user we chose :



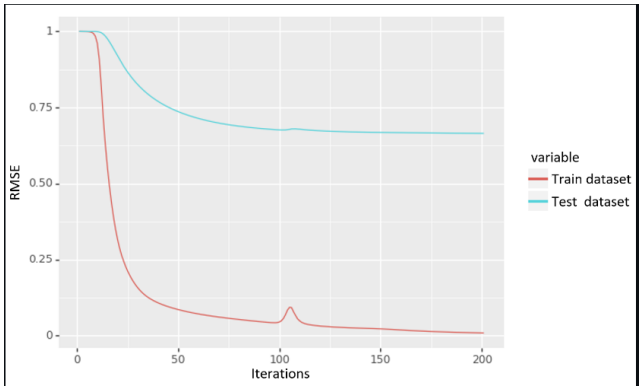
5 Experiments/Results

EM-Algorithm
So when we used the default EM Algorithm on raw data for an example game called Fallout4 we get this graph:



As we can see the EM algorithm does a great job at finding groups of people with similar gaming habits . We can see that some users played the game for few hours . That can mean that these users didn't like the game and lost their interest shortly after starting playing it . The distribution is denser for groups 3 and 4. That shows that the most players are interested in this game so the game would likely be highly rated.

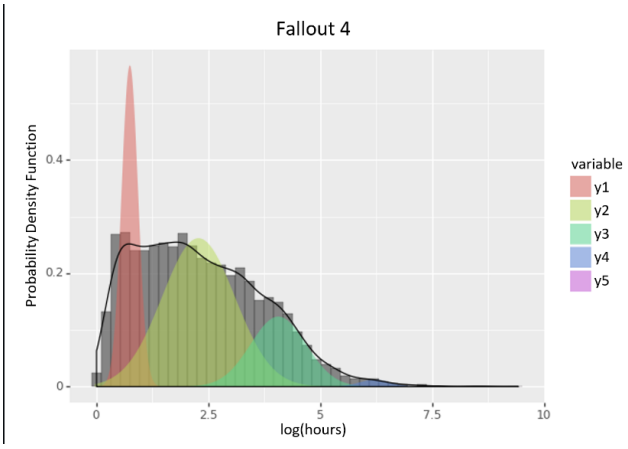
Another thing we wanted to test is the conversion of SVD via gradient descent and RMSE for the train dataset .



We see that RMSE for the train dataset sits around 0.7 but the SVD converges to zero . Also the accuracy on the test dataset stopped improving after the 100th iteration while the RMSE remains around zero. We could improve the accuracy by using more leading components with the trade off being computation time. So for the data we did use we could stop the computation after 100 iterations since the accuracy doesn't improve enough for us to keep it going.

We also have the distribution graph after using the predicted user-item matrix :

We can see that distributions 2 , 3 and 4 fit the data really well . However distribution 1 doesn't fit the data (something that does not concern us because we didn't care about users that played games less than 2 hours) .
Content-Based



When it comes to Content-Based recommendations, we didn't get a chance to experiment extensively with it. The main reason for this was the struggle we had in getting the code to work smoothly. The major roadblock that prevented us from conducting more extensive experiments was the heavy computational load imposed by dealing with four large columns of data. If you follow the sequence of 'option 1' to 'option 2,' inputting a valid user ID, and finally selecting 'option 4,' it puts a significant strain on your computer. However, it's worth mentioning that this approach did produce accurate results, even though it was quite demanding on our machines. We highly recommend trying out the code on your own if your computer can handle it. We've added plenty of comments within the code to guide you through the process. Below, we've included some screenshots of sample outputs for the same user across all content-based scenarios.

```
Recommended games based on Genre and Publisher for user: 6717871
Games:
1. DOOM
2. The International 2019 Battle Pass
3. Alien Swarm
4. Borderlands 2: Ultimate Vault Hunter Upgrade Pack 2
5. Life is Strange: Before the Storm Farewell
6. Life is Strange: Before the Storm Episode 3
7. Sleeping Dogs
8. Claptastic Voyage and Ultimate Vault Hunter Upgrade Pack 2
9. Fight Songs: The Music Of Team Fortress 2
10. PLAYERUNKNOWN'S BATTLEGROUNDS
11. Legacy of Kain: Defiance
12. Just Cause 2: Agency Hovercraft
13. GT Legends
14. Borderlands 2: Assassin Supremacy Pack
15. Euro Truck Simulator 2 - Dutch Paint Jobs Pack
16. Euro Truck Simulator 2 - Prehistoric Paint Jobs Pack
17. Borderlands 2: Tiny Tina's Assault on Dragon Keep
18. Just Cause 2: Multiplayer Mod
19. F.E.A.R.
20. Life is Strange: Before the Storm DLC - Deluxe Upgrade
```

```
Recommended games based on Genre and Popular Tags for user: 6717871
Games:
1. PLAYERUNKNOWN'S BATTLEGROUNDS
2. Euro Truck Simulator 2 - Heavy Cargo Pack
3. Codename CURE
4. Borderlands 2: Tiny Tina's Assault on Dragon Keep
5. Little Inferno
6. Alien Swarm
7. Borderlands 2: Ultimate Vault Hunter Upgrade Pack 2
8. Gotham City Impostors Free to Play
9. Life is Strange: Before the Storm Farewell
10. Life is Strange: Before the Storm Episode 3
11. American Truck Simulator - Wheel Tuning Pack
12. Euro Truck Simulator 2 - Window Flags
13. Euro Truck Simulator 2 - Pirate Paint Jobs Pack
14. Gizmo
15. Tom Clancy's Rainbow Six® Siege - Year 4 Pass
16. Toki Tori
17. Euro Truck Simulator 2 - UK Paint Jobs Pack
18. Euro Truck Simulator 2 - USA Paint Jobs Pack
19. Euro Truck Simulator 2 - High Power Cargo Pack
20. Claptastic Voyage and Ultimate Vault Hunter Upgrade Pack 2
```

```
Recommended games based on Genre, Popular Tags, and Game Details for user: 6717871
Games:
1. DOOM
2. Euro Truck Simulator 2 - USA Paint Jobs Pack
3. F1 2016
4. Zombie Panic! Source
5. Red Faction Guerrilla Steam Edition
6. BioShock Infinite - Season Pass
7. Little Inferno
8. Life is Strange: Before the Storm Farewell
9. Euro Truck Simulator 2 - Window Flags
10. Euro Truck Simulator 2 - Pirate Paint Jobs Pack
11. Gizmo
12. Tom Clancy's Rainbow Six® Siege - Year 4 Pass
13. Toki Tori
14. Euro Truck Simulator 2 - UK Paint Jobs Pack
15. American Truck Simulator - Heavy Cargo Pack
16. Oozi: Earth Adventure
17. Entropy : Zero
18. The International 2019 Battle Pass
19. Mages of Mystralia
20. Half-Life 2: Update
```

```
Recommended games based on Genre, Publisher, Developer, and Game Details for user: 6717871
Games:
1. PLAYERUNKNOWN'S BATTLEGROUNDS
2. BLOCKADE 3D
3. GT Power Pack – Expansion Pack for RACE 07
4. RETRO – Expansion Pack for RACE 07
5. Forward Line
6. STCC The Game 2 – Expansion Pack for RACE 07
7. Magic 2012 Gold Deck Bundle
8. 2260 VR
9. Sanctum: Map Pack 1
10. Strike of Horror
11. Travel Riddles 4-in-1 Bundle
12. Admine
13. Running King
14. BattleRush - Full Pack
15. Zeit²
16. Sphere III - Elite Pack
17. Jets'n'Guns 1+2 Bundle
18. Euro Truck Simulator 2 - Italia
19. Euro Truck Simulator 2 - Special Transport
20. Life is Strange: Before the Storm Episode 2
```

Neural Experiments And Results

Experiments

In the process of developing our Neural Network architecture for the recommendation system, we initially began with implementing custom ideas that we believed would suit our dataset. However, it quickly became apparent that the complexity involved in crafting a highly effective Neural Network was extensive. Therefore, we pivoted to follow a more established architecture that was recommended.

During this transition, we made several changes to adapt the network to our specific requirements. One significant alteration was the replacement of the dot product layer with a concatenation layer. The dot product layer is typically used to measure the similarity between two vectors, which can be computationally intensive. By swapping it out for a concatenation layer, we were able to reduce the computational load, resulting in a more time-efficient model without a significant loss in performance.

Additionally, we introduced new layers such as dropout and dense layers. The dropout layer is a regularization technique that helps prevent overfitting by randomly setting a fraction of input units to zero during training. This encourages the network to learn more robust features that generalize better to unseen data. The dense layers, on the other hand, are fully connected layers that enable the network to learn non- linear combinations of the features. By adding these layers, we aimed to enhance the model's ability to capture complex patterns and relationships in the data, leading to improved recommendation accuracy.

As part of our endeavor to optimize the neural network, we conducted experiments by varying key hyperparameters. Hyperparameters are the configuration settings used to structure the learning process and significantly influence the performance of the neural network.

Learning Rate Adjustment

The learning rate is a critical hyperparameter that affects how much the weights of the network are updated during training. A high learning rate can cause the model to converge too quickly to a suboptimal solution, while a low learning rate can slow down the learning process or cause the training to get stuck.

Experiment: We tested various learning rates with the Adam optimizer, ranging from 0.001 to 0.0001.

Results: Our experiments indicated that a learning rate of 0.0005 provided the best balance between convergence speed and training stability, leading to an improved Mean Squared Error (MSE) on the validation set.

Dropout Rate Variation

The dropout rate determines the percentage of neurons that are randomly deactivated during training, providing a form of regularization.

Experiment: We experimented with dropout rates from 0.1 (10%) to 0.5 (50%).

Results: A dropout rate of 0.3 was found to effectively reduce overfitting while still allowing the network to learn sufficiently complex patterns from the training data.

Number of Neurons in Dense Layers

The number of neurons in dense layers controls the capacity of the network to learn from data.

Experiment: We evaluated the performance of the network with different numbers of neurons, testing configurations that included 64, 128, and 256 neurons in the dense layers.

Results: The network with 128 neurons in each dense layer achieved the best trade-off between model complexity and generalization ability.

Batch Size Optimization

Batch size influences the number of samples that are propagated through the network before the weights are updated.

Experiment: We tested batch sizes of 32, 64, and 128 to observe their impact on the learning dynamics.

Results: A batch size of 64 provided the most consistent decrease in validation loss across epochs, indicating stable learning and better generalization.

Conclusion The hyperparameter tuning experiments played a crucial role in refining our neural network's architecture. By systematically exploring different configurations and observing their impact on the model's performance, we were able to identify the optimal settings that enhanced the accuracy and efficiency of our game recommendation system. These experiments underscore the importance of hyperparameter tuning in the development of robust neural network models.

Results

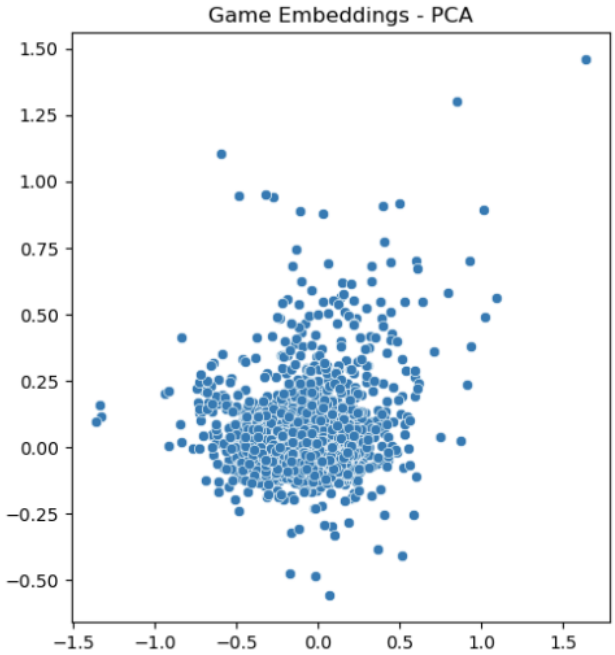
The results of our experiments with the Neural Network have been promising. After incorporating the changes, we observed an improvement in the model's ability to generalize, as evidenced by a decrease in validation loss over successive training epochs. The addition of dropout layers effectively mitigated the overfitting issue, while the dense layers added depth to the network's learning capability. Furthermore, we achieved an acceptable result with the coefficient of determination (R-squared)

value being 0.03, which, while modest, is greater than 0, indicating that the model has predictive power beyond the mean of the dataset.

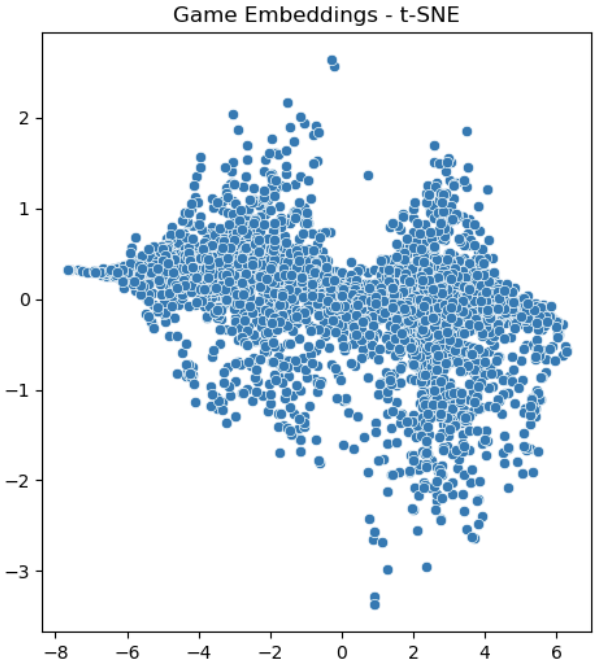
We also leveraged visualizations such as PCA and t-SNE plots to analyze the embeddings learned by the network. These visualizations provided valuable insights into the model's understanding of the user and game relationships, further confirming the efficacy of our architectural decisions.

In conclusion, the adjustments made to the Neural Network architecture were necessary steps to tailor the model to our dataset's unique characteristics. Despite the challenges, the alterations and additions to the model's layers have contributed positively to the overall performance of the recommendation system.

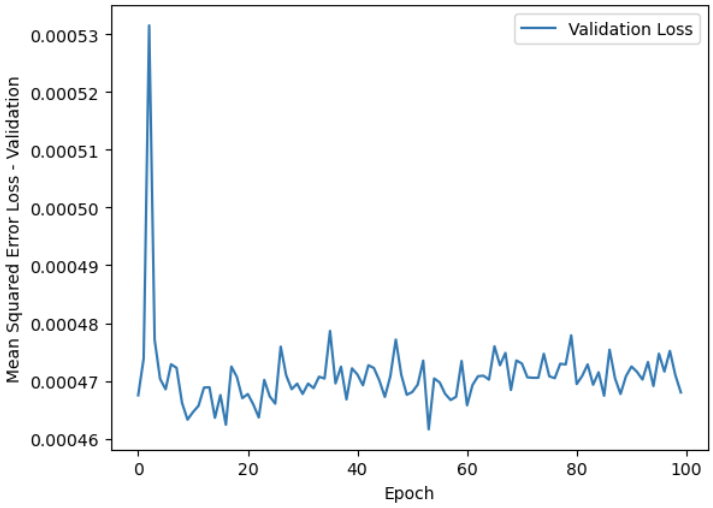
Visualization of the Final Neural Network Embeddings and Performance



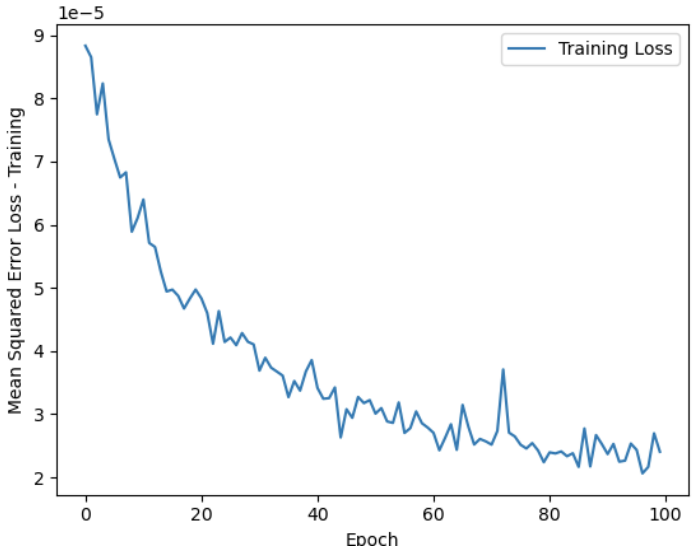
The PCA visualization of game embeddings shows a dense cluster with outliers, indicating that while many games share similar features, there are some with unique characteristics.



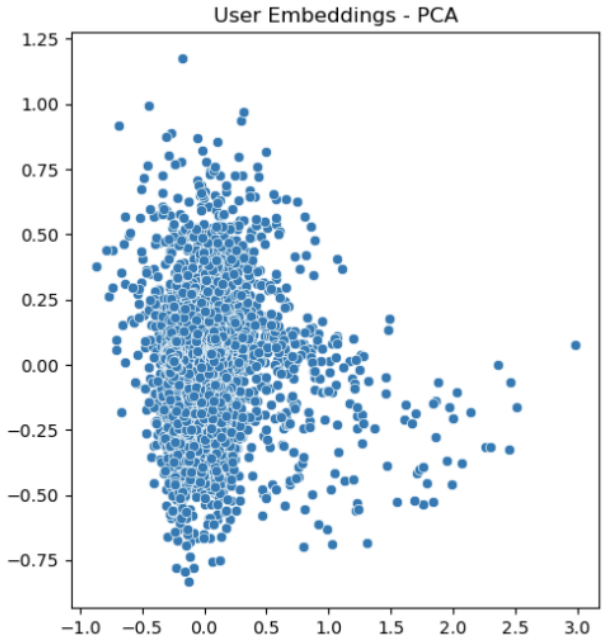
The t-SNE plot reveals distinct clusters of games, suggesting the presence of groups with similar properties within the high-dimensional embedding space.



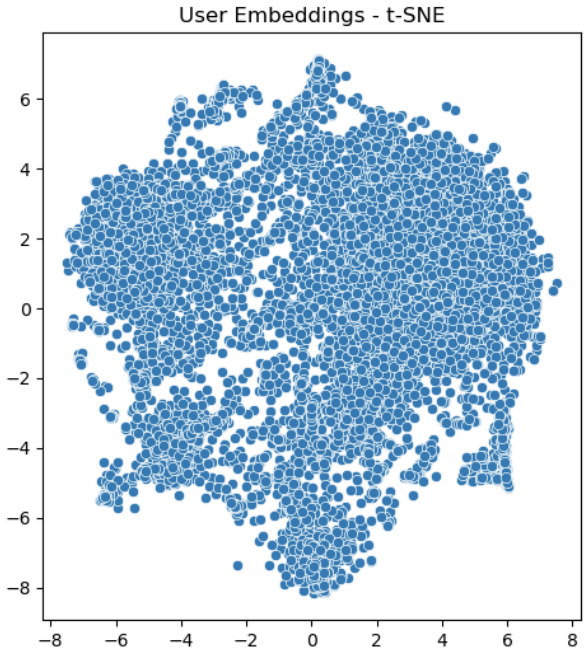
The validation loss over epochs fluctuates, which may indicate overfitting or the need for learning rate optimization for better generalization.



The training loss graph shows fluctuation across epochs, suggesting potential areas for hyperparameter tuning and model stabilization.



The PCA plot for user embeddings indicates a wide range of user behaviors and preferences, with a high-density region and some dispersed points.



The t-SNE visualization for user embeddings uncovers various clusters, which could represent distinct user groups with similar gaming interests and behaviors.

In conclusion, the adjustments made to the Neural Network architecture were necessary steps to tailor the model to our dataset's unique characteristics. Despite the challenges, the alterations and additions to the model's layers have contributed positively to the overall performance of the recommendation system.

6 Conclusion And Future Work

6.1 Conclusion

In summary, our game recommendation system integrates three primary algorithms: Collaborative Filtering, Content-Based Filtering, and Neural Networks. Each algorithm has its own weaknesses and strengths, contributing to a robust and versatile recommendation engine.

Collaborative Filtering, particularly when employing Gradient Descent, emerged as the overall best-performing algorithm in terms of efficiency. While its recommendations might not be as sharply targeted as those generated by Content-Based Filtering, its ability to efficiently handle large datasets made it the most practical choice for our system.

Content-Based Filtering, on the other hand, achieved the highest accuracy in our experiments. It excels in providing highly targeted recommendations by analyzing specific game features. However, its computational intensity proved to be a substantial burden on system resources, especially as the number of features increased. This trade-off between accuracy and efficiency highlighted the algorithm's limitations in a real-world application where quick response times are essential.

The most resource-intensive aspect of our project was training the Neural Network. As anticipated, the process was both time-consuming and costly in terms of computational resources.

Overall, the selection of the most appropriate algorithm depends on the specific needs of the system in question. For our Game Recommendation System, Collaborative Filtering struck the best balance between efficiency and effectiveness, making it the cornerstone of our approach. Despite the challenges, our comprehensive system takes significant strides towards enhancing user experience in the vast gaming landscape, saving users time and connecting them with games they are likely to enjoy.

6.2 Future Work

Moving forward, there are several areas for advancement and refinement of our Game Recommendation System. A primary focus for future work will be to delve deeper into optimization techniques for Content-Based Filtering. By exploring and implementing more sophisticated methods, we aim to enhance the algorithm's efficiency without sacrificing the high degree of accuracy it currently achieves. This may involve investigating dimensionality reduction techniques or more advanced similarity measures that could streamline the feature analysis process.

Another exciting direction would be the development of a custom neural network tailored explicitly to the needs of our dataset and the specific requirements of game recommendation. Given the necessary resources and time, a custom neural network architecture could be designed to optimize performance, leveraging the latest advancements in deep learning research. This project would involve a thorough examination of the various layers, activation functions, and network structures to create a model that is both highly accurate and resource-efficient.

Additionally, extending our team to include additional in neural networks and optimization could significantly accelerate progress in these areas. With a dedicated group of specialists, we could push the boundaries of what our Game Recommendation System can achieve.

7 Contributions

This project was a collaborative effort between Trifon and Ioannis, with both team members contributing significantly to various aspects of the Game Recommendation System.

7.1 Collaborative Contributions

Both Trifon and Ioannis made substantial contributions to the collaborative components of the recommendation system. They worked together on the preprocessing and cleaning of the dataset, a critical step that set the foundation for the effective functioning of the Collaborative Expectation-Maximization (EM) algorithms.

7.2 Trifon's Specialization in Content-Based Filtering

After their joint efforts on the collaborative aspects, Trifon took the lead on the Content-Based filtering part of the project. He meticulously tailored the recommendation system to align with users' preferences and behaviors by leveraging the features inherent to the games. In addition to his analytical tasks, Trifon also applied his skills in UI/UX design, enhancing the Content-Based recommendation module's interactivity and user engagement.

7.3 Ioannis's Specialization in Neural Networks

Concurrently, Ioannis delved into the neural network domain of the recommendation system. He refined the data specifically for neural processing and employed his deep understanding of neural architectures to construct an effective model. His contributions were not only technical but also extended to integrating the neural network module with the existing user interface, facilitating a smooth and intuitive experience for personalized game recommendations.

7.4 Summary of Collaborative and Individual Contributions

The combined efforts of Trifon and Ioannis in the initial stages of collaboration, followed by their specialized focus—Trifon on Content-Based filtering and Ioannis on neural network development—were instrumental in building a comprehensive and multifaceted recommendation system. Their distinct yet complementary skills contributed significantly to the project's success. Both Trifon and Ioannis collaborated on writing parts of the LaTeX report, ensuring a comprehensive and well-documented presentation of their work. They made efforts to distribute the workload evenly, with each member contributing approximately 50 of the time and resources required for the project.

7.5 Conclusion

The combined efforts of Trifon and Ioannis in different areas of the project were instrumental in the successful development of the Game Recommendation System. Their collaboration highlights the importance of teamwork and diverse skill sets in tackling complex machine learning projects.

References/Bibliography

1. Oehm, D. (n.d.). Steam Game Recommendations. [Online]. Available from `https://www.kaggle.com/code/danieloehm/steam-game-recommendations/notebook`
   - (Citation for the collaborative recommender using the EM algorithm.)
2. DataCamp. (n.d.). Recommender Systems in Python: Beginner Tutorial. [Online]. Available from `https://www.datacamp.com/tutorial/recommender-systems-python`
   - (Citation for the content-based recommender system.)
3. Maruti Techlabs. (n.d.). Recommendation Engine - What, Why & How? [Online]. Available from `https://marutitech.com/recommendation-engine-benefits/`
   - (Citation for understanding what a recommender system is and the different types.)
4. Towards Data Science. (n.d.). Neural Collaborative Filtering. [Online]. Available from `https://towardsdatascience.com/neural-collaborative-filtering-96cef1009401`
   - (Citation for Neural Collaborative Filtering.)
5. NVIDIA Developer Blog. (n.d.). Using Neural Networks for Your Recommender System. [Online]. Available from `https://developer.nvidia.com/blog/using-neural-networks-for-your-recommender-system/`
   - (Additional citation for Neural Collaborative Filtering.)
6. `https://dl.acm.org/doi/pdf/10.1145/3383313.3412488`

7. https://arxiv.org/ftp/arxiv/papers/1712/1712.07525.pdf
8. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10247320
9. https://www.researchgate.net/publication/318113063_Recommender_Systems_for_Personalized_Gamification
10. https://www.mdpi.com/2079-9292/11/1/141